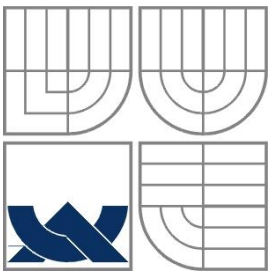
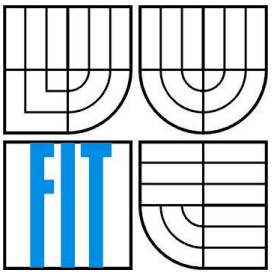


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

VYLEPŠENÍ FORMÁTU JPEG

IMPROVEMENTS OF JPEG FORMAT

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

RADEK LIŠKA

VEDOUCÍ PRÁCE
SUPERVISOR

ING. DAVID BAŘINA

BRNO 2011

Abstrakt

Tato práce se zabývá algoritmy ztrátové komprese statického obrazu používanými standardem JPEG. Jsou zde podrobně představeny jednotlivé kroky komprese podle tohoto standardu. Mým cílem je představení a implementace vylepšení, která by vedla ke zvýšení kompresního poměru při zachování struktury standardu. Tato implementace je v závěru práce srovnána se standardní implementací a jsou navržena další možná vylepšení.

Abstract

This thesis deals with lossy still image compression algorithms used in JPEG standard. All compression steps according to this standard are described in detail. My objective is to introduce and implement improvements that would result in increased compression ratio while adhering to the structure of the standard. At the end of the thesis, the implementation is compared to standard implementation and additional possible improvements are proposed.

Klíčová slova

Ztrátová komprese obrazu, JPEG, JPEG-Plus, DCT, linearizace, Mortonův průchod, pyramidový průchod, entropické kódování, Recursive Range Reduction, Golomb-Riceovy kódy, Fibonacciho kódy

Keywords

Lossy image compression, JPEG, JPEG-Plus, DCT, linearization, Morton scan, pyramid scan, entropy coding, Recursive Range Reduction, Golomb-Rice codes, Fibonacci codes

Citace

Radek Liška: Vylepšení formátu JPEG, bakalářská práce, Brno, FIT VUT v Brně, 2011

Vylepšení formátu JPEG

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Davida Bařiny. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Radek Liška
16. května 2011

Poděkování

Chtěl bych poděkovat svému vedoucímu Ing. Davidu Bařinovi, který mi poskytl odbornou pomoc při vypracovávání této práce a při četných konzultacích mě vždy správně směroval.

Obsah

1 Úvod

I v dnešní moderní době, kdy jsou cenově běžně dostupné disky s kapacitou v řádech terabajtů, internetová připojení s rychlostí v řádech megabitů, a kdy se výkon procesorů posuzuje nejen podle taktu, ale i podle počtu jader, je stále v počítačovém světě využíváno komprese a to zvláště v oblasti multimédií. Ať už se budeme chtít dívat na film, poslouchat hudbu či brouzdat po internetu, všude narazíme na komprimovaná data. Zvláště při prohlížení rastrových obrázků na internetu se už jen vzácně setkáme se souborem v nekomprimovaném formátu. Tato práce se zabývá prozkoumáním jednoho z nejrozšířenějších algoritmů ztrátové komprese obrazu, který je definovaný ve standardu JPEG, a popsáním existujících metod, které by mohly vést ke zvýšení jeho účinnosti. Tyto metody jsou ve druhé části práce naimplementovány a je zkoumán jejich účinek ve srovnání s metodami používanými ve standardu.

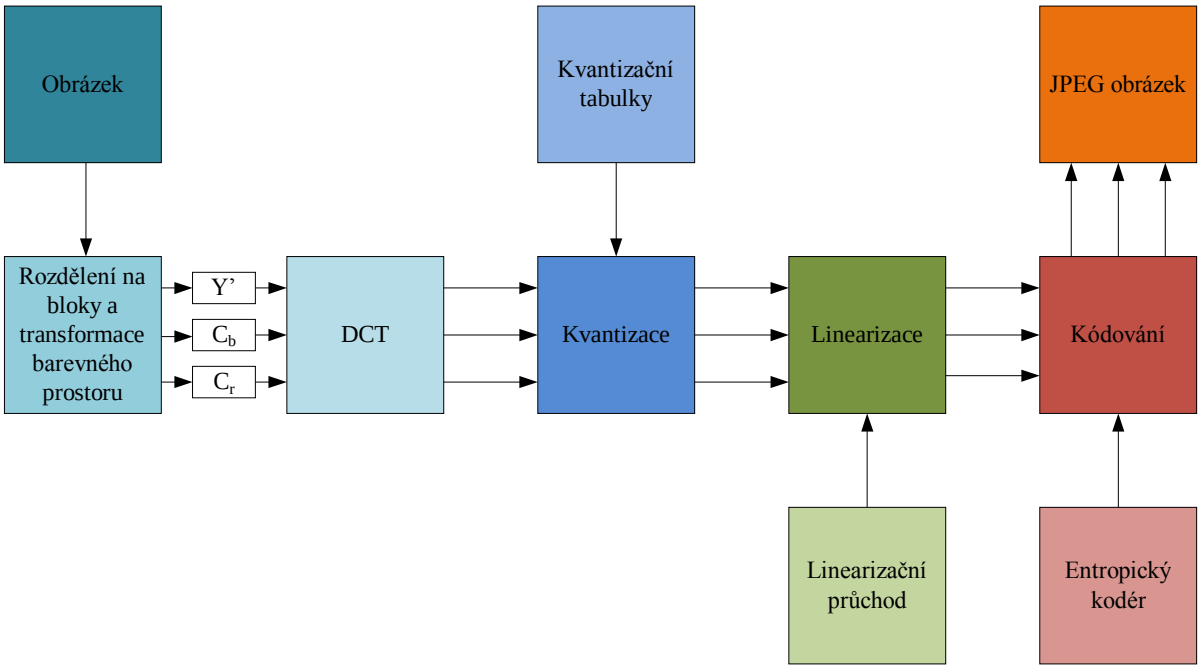
Ve druhé kapitole jsou rozebrány jednotlivé části komprese obrazu dle standardu JPEG. Je zde popsána transformace barevného prostoru z RGB do $Y'C_bC_r$, zpracování jednotlivých bloků obrazu pomocí diskrétní kosinové transformace, kvantizace takto transformovaných dat za pomoci kvantizačních tabulek, jejich linearizace do jednorozměrného proudu a podrobný popis postupů kódování těchto proudů na výstup.

Ve třetí kapitole je popsána první část navrhovaných vylepšení – navýšení velikosti bloku a přidání dalších linearizačních průchodů – a ve čtvrté kapitole jsou popsány nově navrhované entropické kodéry.

V následující kapitole se nachází srovnání časové náročnosti komprese různých velikostí bloků, poté jsou srovnány jednotlivé oblasti vylepšení a nakonec je srovnáno kompletní řešení. V poslední kapitole je zhodnocen přínos práce, zmíněny další provedené experimenty a načrtnut další směr vývoje programu.

2 Průběh komprese JPEG

Ztrátová komprese obrazu dle standardu JPEG [8][10][12] sestává z pěti pevně daných kroků. Zpracovávaný obraz je z důvodu snížení výpočetní náročnosti rozdělen na bloky o velikosti 8×8 a je provedena transformace barevného prostoru z RGB na $Y'C_bC_r$. Na každý blok je aplikována diskrétní kosinová transformace, výsledné bloky frekvenčních koeficientů jsou poté kvantizovány kvantizačními tabulkami pro zlepšení možností jejich zakódování, linearizovány linearizačním průchodem a nakonec entropickým kodérem zakódovány na výstup. Při dekódování je proces opačný, na ztrátách na kvalitě a vznikajících artefaktech se nejvíce podílí kvantizace během kompresního kroku, jelikož zaokrouhlením dochází k nereverzibilní ztrátě informací. Kvantizace ale zároveň nejvíce zvyšuje kompresi vstupního obrazu.



Obrázek 2.1: Schéma kodéru dle standardu JPEG

2.1 Transformace barevného prostoru

Velké množství digitálních obrazových dat určených k zobrazení na displeji je uloženo v barevném modelu RGB. RGB reprezentuje obrazový bod pomocí úrovně červené, zelené a modré složky (z anglických názvů těchto barev Red, Green, Blue pochází i jeho zkratka). V každé komponentě se nachází podobné množství informace. Pro ztrátové kódování obrazu jsou ale vhodnější jiné modely, protože v RGB se ve všech třech komponentách projevuje jas, na který je lidský zrak nejvíce citlivý.

V barevném modelu $Y'C_bC_r$ (někdy zapisováno jako $Y'C_BC_R$) se všechny tři složky na výsledném vzhledu obrazového bodu, jak jej vnímá lidské oko, nepodílejí stejně, ač jde jen o odlišné kódování shodných RGB informací. Složka Y' je jasová (jinak také luma), zatímco C_b a C_r složky jsou barvonosné (chroma), na tyto je zrak citlivý méně než na jasovou složku. Proto je možné z barvonosných složek použít menší datový objem při zachování nebo menším snížení výsledné

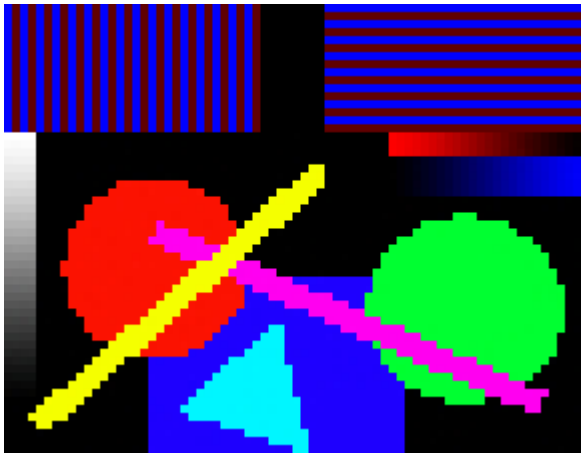
vnímané kvality obrazu než při obdobné ztrátě informací v jasové složce. V JPEG standardu jsou určeny rovnice na převod mezi barevným modelem RGB a Y'C_bC_r:

$$\begin{aligned} Y' &= 0{,}29900 \times R + 0{,}58700 \times G + 0{,}11400 \times B \\ C_b &= -0.16874 \times R - 0.33126 \times G + 0.50000 \times B + 128 \\ C_r &= 0.50000 \times R - 0.41869 \times G - 0.08131 \times B + 128 \end{aligned} \quad (1)$$

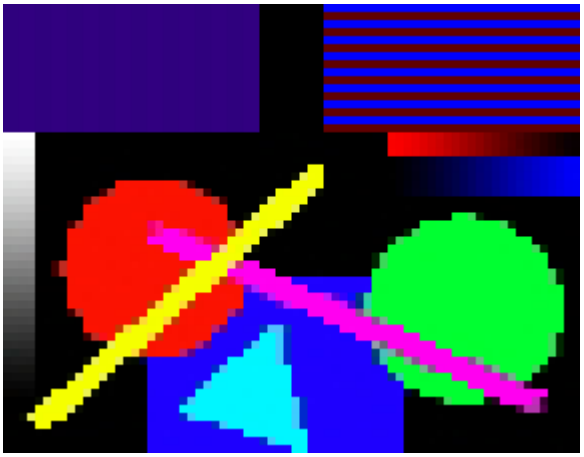
Počítá se s 8bitovou hloubkou jedné barevné komponenty, tzn. vstupní hodnoty RGB i výstupní hodnoty Y'CbCr jsou v rozsahu 0–255. Rovnice pro zpětný převod z Y'CbCr do RGB:

$$\begin{aligned} R &= Y' + 1.40200 \times (C_r - 128) \\ G &= Y' - 0.34414 \times (C_b - 128) - 0.71414 \times (C_r - 128) \\ B &= Y' + 1.77200 \times (C_b - 128) \end{aligned} \quad (2)$$

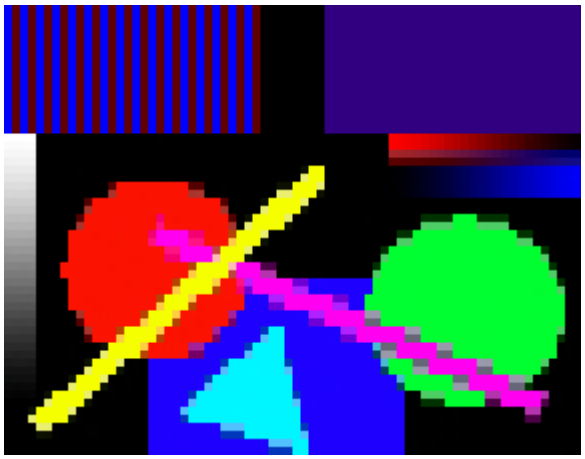
Zvláště při vyšších kompresních poměrech se často využívá podvzorkování (subsampling) chromy, kdy se sníží rozlišení této komponenty. Nejběžnější podvzorkování jsou o velikosti vzorku 2×1 (snížení horizontálního rozlišení na polovinu), 1×2 (snížení vertikálního rozlišení na polovinu) a 2×2 (snížení obou rozměrů komponenty na polovinu). Vliv podvzorkování na barevnou informaci ilustrují obrázky 2.2.a až 2.2.d [7]:



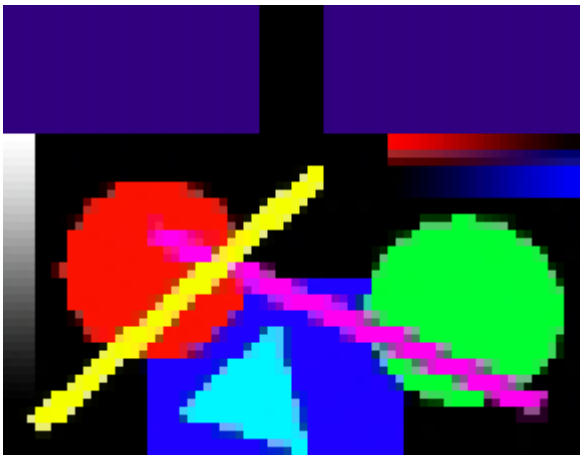
Obrázek 2.2.a: Vzorek 1×1 (bez podvzorkování)



Obrázek 2.2.b: Vzorek 2×1



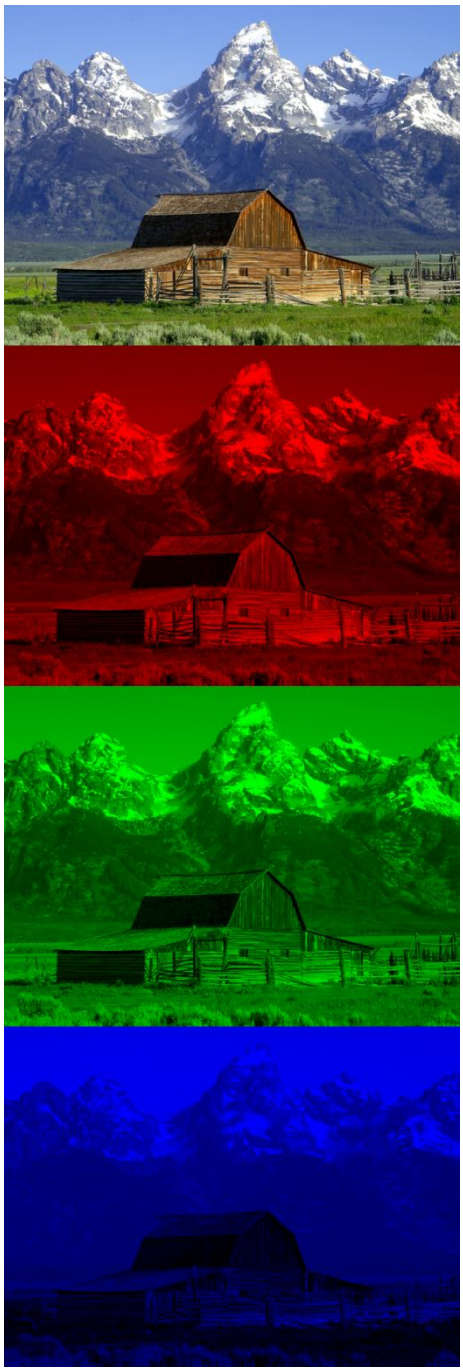
Obrázek 2.2.c: Vzorek 1×2



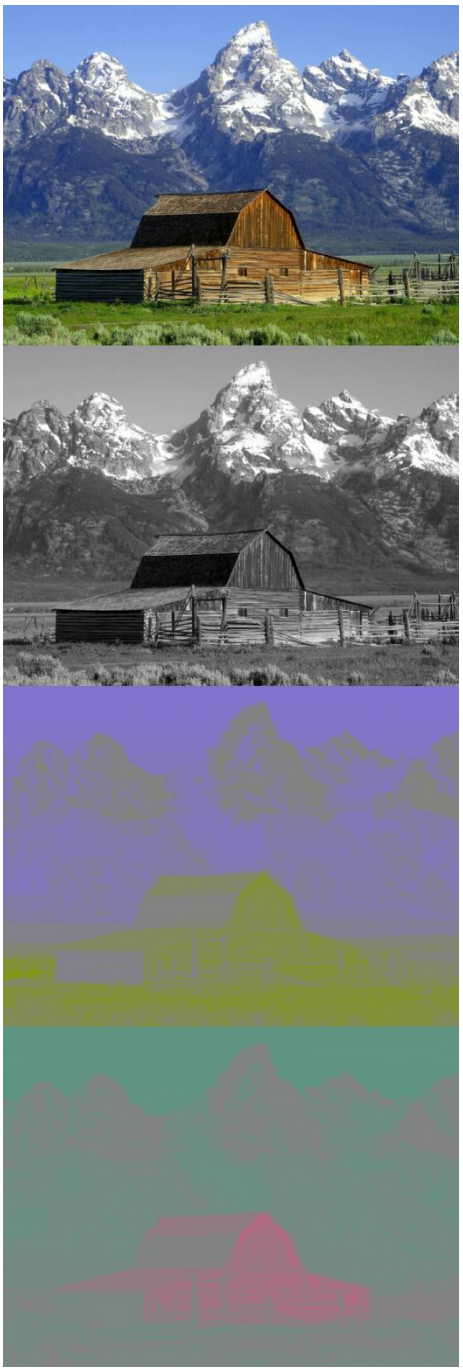
Obrázek 2.2.d: Vzorek 2×2

Na obrázku 2.3.a se nachází rozdělení RGB obrázku na jednotlivé komponenty - červenou, zelenou a modrou, na obrázku 2.3.b je rozdělení Y'C_bC_r obrázku na lumu a dvě chromy. Na první

pohled je vidět, že lidským okem snáz vnímaný jas se v modelu RGB projevuje ve všech třech složkách, zatímco v modelu $Y'C_bC_r$ je koncentrován v lumě.



Obrázek 2.2.a: komponenty obrazu v RGB



Obrázek 2.2.b: komponenty obrazu v $Y'C_bC_r$

2.2 DCT

DCT (diskrétní kosinová transformace) [2] převádí signál z prostorové (dvourozměrná matice pixelů) do frekvenční domény (dvourozměrná matice frekvencí a jejich energií). Nejčastěji používaná DCT-II (1), běžně zvaná jen „DCT“, je tvořená následující rovnicí:

$$X_k = \sum_{n=0}^{N-1} x_n \cos\left[\frac{\pi}{N}\left(n+\frac{1}{2}\right)k\right] \quad k = 0,\ldots,N-1 \tag{3}$$

Inverzní transformace k DCT-II je DCT-III (2) násobená 2/N („IDCT“). Samotná DCT-III:

$$X_k = \frac{1}{2}x_0 + \sum_{n=1}^{N-1} x_n \cos\left[\frac{\pi}{N}n\left(k+\frac{1}{2}\right)\right] \quad k = 0,\ldots,N-1 \tag{4}$$

Na bloku se provádí dvourozměrná DCT, která se provede pomocí aplikace série jednorozměrných DCT na řádky a poté na sloupce. DCT koncentruje nejvíce energie na nízkých frekvencích obrazu, zejména v prvním elementu tvořícím DC koeficient. Zkratka DC znamená Direct Current - stejnosměrný proud - frekvence tohoto koeficientu je vždy nulová, jeho hodnota je rovna průměru hodnot celého bloku. Ostatní elementy jsou AC koeficienty (Alternating Current = střídavý proud). Vysoké frekvence obsahují informace o hranách (ostrých barevných přechodech). Čím vyšší index v horizontálním, resp. vertikálním směru, tím vetší jsou horizontální, resp. vertikální frekvence, jejichž koeficienty jsou na daném indexu uloženy. Ukázka DCT aplikovaného na jeden blok dat je n obrázku 2.3. Při vynulování frekvencí AC koeficientu a využití pouze hodnoty DC koeficientu vznikne obrázek, který bude vypadat jako stylizace původního obrázku do jednobarevných 8×8 dlaždic.

162	162	162	162	162	164	160	158
162	162	162	162	162	164	160	158
162	162	162	162	162	157	163	155
161	161	161	161	161	155	158	156
162	162	162	162	162	161	160	158
156	156	156	156	156	159	161	158
163	163	163	163	163	158	159	156
160	160	160	160	160	159	155	157

DCT →

2078	32	16	-1	1	-1	-30	37
46	-1	2	-19	7	12	-16	13
-29	-2	-6	6	-2	-2	8	-7
13	5	6	5	-2	-5	0	2
-6	-5	-1	-6	6	4	-5	0
7	3	-6	-0	-8	3	6	2
-10	0	11	6	6	-9	-3	-4
8	0	-9	-6	-3	8	1	3

Obrázek 2.3: Hodnoty zpracovávaného bloku před a po DCT

2.3 Kvantizace

DCT je v podstatě bezztrátová transformace, případné ztrátě informací se dá vyhnout zvýšením přesnosti reprezentace čísel. Pokles kvality ve ztrátové kompresi obrazu způsobuje kvantizace. Kvantizace probíhá tak, že se matice frekvenčních koeficientů vzniklých při DCT vydělí pomocí kvantizační tabulky. Pro každou obrazovou složku existuje zvláštní tabulka, obvykle jedna pro jasovou složku a jedna pro barvonosné složky. Koeficienty kvantizační tabulky jsou zpravidla přímo úměrné velikosti frekvence na dané pozici, tedy zvyšují se podél hlavní diagonály. Existují ale i výjimky, např. tabulky upravené pro lidský zrak (HVS - Human Visual System) [6], které mají definovanou váhu jednotlivých kvantizačních koeficientů tak, aby pro lidské vnímání vznikl rozdíl nižší, než který se projeví na datovém toku. V příkladu tabulek uvedených v JPEG standardu lze vidět důraz na snížení objemu informace v barvonosných složkách a ve vyšších frekvencích všech tří barevných složek. Hodnoty koeficientů kvantizačních tabulek je možné škálovat pomocí uživatelského nastavení kvality. Toto nastavení je přítomné ve všech rozšířených obrazových editorech podporujících formát JPEG a je pro uživatele zpravidla jediný způsob, jak ovlivnit kvalitu

a datovou velikost výsledného obrazu. Standardní tabulky i HVS tabulky jsou předmětem experimentů této práce. Vliv kvantizace na snížení objemu dat je patrný z obrázku 2.4.

2078	32	16	-1	1	-1	-30	37
46	-1	2	-19	7	12	-16	13
-29	-2	-6	6	-2	-2	8	-7
13	5	6	5	-2	-5	0	2
-6	-5	-1	-6	6	4	-5	0
7	3	-6	-0	-8	3	6	2
-10	0	11	6	6	-9	-3	-4
8	0	-9	-6	-3	8	1	3

Q →

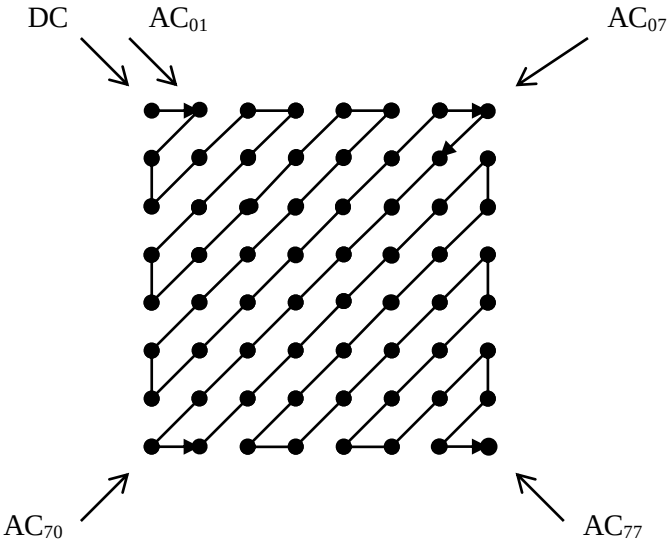
126	3	2	0	0	0	-1	1
4	0	0	-1	0	0	0	0
-2	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Obrázek 2.4: Hodnoty zpracovávaného bloku před a po kvantizaci

2.4 Linearizace

Blok transformovaný pomocí DCT po kvantizaci stále tvoří dvourozměrnou matici koeficientů. Jelikož většina kompresních algoritmů byla vytvořena pro zpracování jednorozměrného proudu dat a ve vícerozměrných proudech dat žádné spojitosti detekovat nedokáže, je třeba data nejprve převést do jednorozměrné podoby, než budou zakódována. V kvantizovaném bloku jsou nejvyšší hodnoty v DC koeficientu a jeho nejbližším okolí a poté podél hlavní diagonály klesají a v oblasti vysokých frekvencí se již většinou nachází jen sekvence nul. Načítáním koeficientů po řádcích by se posloupnosti nul přerušovaly, jak by se přecházelo mezi oblastmi s nízkými a s vysokými frekvencemi, proto je potřeba použít sofistikovanější algoritmus, aby posloupnosti zůstaly zachovány.

Tento úkol v JPEG standardu plní prvek linearizace. Ve standardu se využívá cik-cak průchod (na obrázku 2.5). Cik-cak průchod začíná v matici v DC koeficientu a postupně prochází po hlavní diagonále a čte po řádcích kolmo na ni. Tím zůstává zachována posloupnost frekvencí a zpravidla i sekvencí nul.



Obrázek 2.5: Schéma cik-cak průchodu

2.5 Entropický kodér

Výsledný proud linearizovaných obrazových dat je třeba nějakým způsobem zakódovat do souboru. V JPEG standardu se využívá Huffmanův kodér [5] pro jeho nízkou hardwarovou náročnost a dobrou účinnost. Další možností popisovanou v JPEG standardu je aritmetický kodér, ale šíření jeho implementace v tomto obrazovém formátu bylo omezeno kvůli patentovému zatížení technik aritmetického kódování. Aritmetický kodér poskytuje zlepšení kompresního poměru o 5 až 10 % ve srovnání s Huffmanovým kodérem, navýšení kompresního a dekompresního času je ale výraznější, zvláště na starším hardwaru, a tvoří další důvod problematického prosazování aritmeticky kódovaných obrázků.

2.5.1 Huffmanův kodér

Huffmanův kodér má definovaný způsob kódování obou druhů koeficientů. DC koeficienty se kódují odděleně od AC koeficientů. Sousední DC koeficienty v blocích spolu výrazně korelují, proto se DPCM kódováním (Delta Pulse Code Modulation, také známé jako rozdílové kódování) snižuje hodnota, kterou je třeba zakódovat na výstup. DPCM kódování se provádí tak, že se od aktuální hodnoty DC koeficientu a barevné komponenty odečítá hodnota DC koeficientu stejné barevné komponenty z předchozího bloku. Jelikož jejich hodnoty bývají v mnoha případech podobné, zakódovat jejich rozdíl je výhodnější než kompletní hodnotu. Bloky se zpracovávají po řádcích zleva doprava shora dolů, první DC koeficient na novém řádku má implicitního předchůdce hodnoty 0. Na výstup se nejprve ukládá Huffmanovým kódováním zakódovaná diferenční kategorie, poté hodnota DC koeficientu. Diferenční kategorie říká, na kolika bitech bude uložená jeho hodnota.

AC koeficienty se před zakódováním zpracovávají variantou RLE kódování (Run-Length Encoding), jehož každý kód tvoří dvě složky. První složka reprezentuje počet za sebou následujících AC nul v linearizovaném proudu (`Run` – běh), druhá složka reprezentuje diferenční kategorii (`Size` - velikost) prvního nenulového AC koeficientu a – pokud je `Size` větší než 0 – za kódem následuje samotná hodnota AC koeficientu. Každá ze složek je na 4 bitech, dohromady každé kódové slovo tvoří jeden bajt. Existují dvě speciální hodnoty kódu, jedna je `Run=15, Size=0` a nazývá se ZRL, druhá je `Run=0, Size=0` a nazývá se EOB. ZRL znamená Zero Run-Length a označuje běh 16 nul. EOB znamená End Of Block a značí konec bloku. EOB signalizuje, že všechny zbývající AC koeficienty jsou nulové a dekodér je tedy dokáže při dekompresi do bloku doplnit, aniž by potřeboval další data uložená v souboru. Tímto lze dosáhnout velmi výrazného zvýšení účinnosti kódování.

Příklad pro závěr sekvence AC koeficientů o hodnotách `0, 0, 0, 0, 0, 0, 1, 3, 0, 0, 0:` `(6,1)[1](0,2)[11](3,0)(0,0)`. V kulatých závorkách jsou RLE kódy pro sekvence nul, zatímco v hranatých závorkách jsou přímo bitové reprezentace zakódovaných čísel. Jelikož v každé diferenční kategorii jsou z kladných čísel jen ta, která k reprezentaci přímo potřebují stejný počet bitů, tedy nejvýznamnější bit nastavený na jedničku, je možné druhou polovinu binárních řetězců použít pro zakódování záporných čísel. Záporná čísla lze zakódovat jako kladná a poté všechny bity obrátit (30_b = 11110, -30_b = 00001).

Jádro Huffmanova kódování probíhá tak, že se symboly vytvořené pomocí RLE kódování v prvním průchodu seřadí podle četnosti jejich výskytu a nad nimi se sestaví binární strom. Dva kódy, uzly stromu, či jejich kombinace s nejnižším součtem četností vytváří nový uzel stromu, který má četnost výskytu rovnou součtu jejich. Jakmile jsou všechny počáteční hodnoty zpracovány, ve druhém průchodu se všem symbolům přiřazují kódová slova dle cesty vedoucí k nim ve stromě

(např. volba levé větve stromu odpovídá bitové ‚0' a volba pravé bitové ‚1'). Platí tedy, že čím častěji se daný symbol vyskytuje, tím je jeho kód kratší. Doporučené Huffmanovy kódy pro jednotlivé možné hodnoty jsou uvedeny ve standardu.

2.5.2 Aritmetický kodér

Princip aritmetického kodéru tkví ve snaze reprezentovat vstupní řetězec symbolů jako binární reprezentaci číselného zlomku z intervalu <0; 1). Pokud máme abecedu o třech znacích – a, b, c – a pravděpodobnosti jejich výskytu jsou po řadě 0,5, 0,25 a 0,25, jsou intervaly pro zápis jednoho jejich výskytu po řadě <0; 0,5), <0,5; 0,75), <0,75; 1). Při kódování výskytu symbolu se celkový interval rekurzivně vnořuje do intervalu právě zakódovaného symbolu. Tedy při kódování zprávy „ab" se po zakódování „a" zpřesňuje celkový interval na <0; 0,5) a intervaly jednotlivých symbolů na <0; 0,25), <0,25; 0,375), resp. <0,375; 0,5). 2 znaky dlouhou zprávu „ab" lze tedy zakódovat jako číslo 0,25.

Ve standardu JPEG se používá adaptivní binární aritmetický kodér – binární znamená, že jím zpracovávaná abeceda obsahuje dva symboly. Nezpracovává tedy vstup jako posloupnost bajtů, jak je časté u jiných entropických kodérů, ale jako řadu bitových jedniček a nul. Adaptivní znamená, že nemá pevně nastavené konstantní pravděpodobnosti výskytu symbolů jako například u Huffmanova kodéru tím, že by se před zakódováním dat nejprve v předcházejícím průchodu zjistila frekvence výskytů jednotlivých symbolů. V adaptivním binárním aritmetickém kodéru dle standardu JPEG je používán stavový automat o 113 stavech ve formě tabulky pravděpodobností výskytu binárních symbolů; mezi těmito stavy kodér plynule přechází po každém zakódovaném symbolu. Vzhledem k tomu, že k optimálnějšímu kódování se v některých stavech vyměňují hodnoty pravděpodobnosti mezi symboly, nenazývají se kódované symboly 0 a 1, ale MPS (More Probable Symbol – pravděpodobnější symbol) a LPS (Less Probable Symbol – méně pravděpodobný symbol). Ve standardu se používá jedna proměnná pro uložení pravděpodobnostního intervalu a druhá jako buffer pro výstupní data. Jelikož intervaly jsou v paměti počítače reprezentované datovým typem omezené délky, dochází při aritmetickém kódování i dekódování po určitém počtu kroků k renormalizaci – pokud proměnná s uloženým intervalem dosáhne kritické hodnoty, vypíšou se data uložená v bufferu a hodnota obou proměnných se upraví na vhodnější hodnoty, tímto je možné generovat binární zlomek s teoreticky neomezenou přesností.

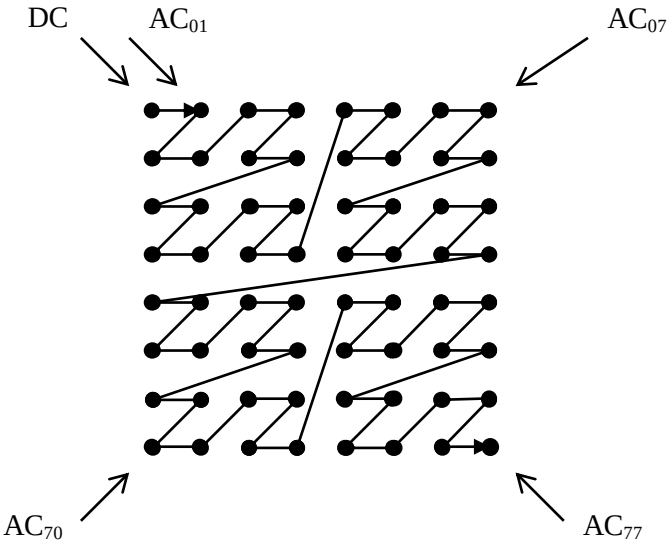
Aritmetický kodér zpravidla dosahuje vyšší účinnosti komprese než Huffmanův kodér díky tomu, že jeden symbol nemusí reprezentovat na celém počtu bitů. Tato výhoda je kompenzována vyšší hardwarovou náročností. V testovací aplikaci aritmetický kodér není naimplementován.

3 Vylepšení linearizačních průchodů a velikosti bloku

S dnešním výkonnějším hardwarem lze pro zvýšení účinnosti použít bloky 16×16 namísto současně používaných 8×8. S blokem 16×16 může vzrůst počet a délka posloupností nulových koeficientů, které díky vhodně zvolenému linearizačnímu průchodu dokáže entropický kodér úsporně zakódovat. Ve standardu JPEG je použit pouze průchod cik-cak, žádné další průchody ve standardu nejsou definovány. Cik-cak průchod i všechny linearizační průchody navrhované v této práci lze použít pro libovolnou velikost bloku, která odpovídá mocnině dvou.

3.1 Mortonův průchod

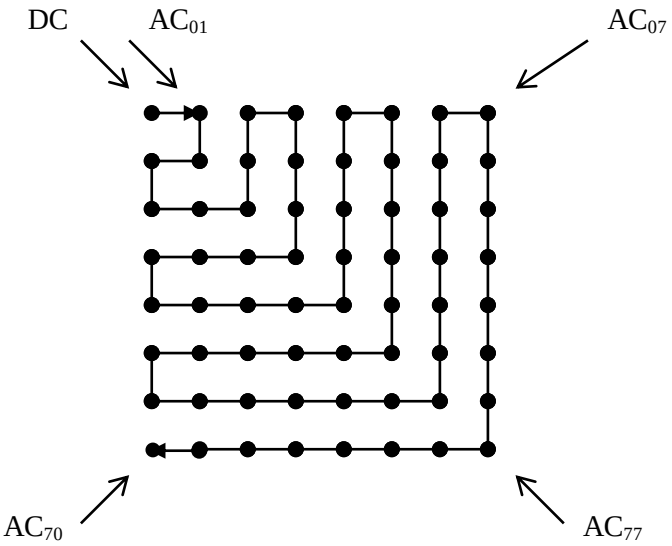
Mortonův průchod [11][15] (na obrázku 2.4) se již ve ztrátové kompresi obrazu používá v kompresních technikách využívajících vlnkové transformace. Jeho tvar se podobá písmenu Z a je rekurzivní povahy.



Obrázek 3.1: Schéma Mortonova průchodu

3.2 Pyramidový průchod

Pyramidový průchod (na obrázku 2.5) je představen v návrhu JPEG-Plus [13]. Návrh jej předkládá jako způsob, jak progresivně kódovat různá rozlišení obrazu hierarchickým postupem.



Obrázek 3.2: Schéma pyramidového průchodu

4 Vylepšení kodéru

JPEG standard zahrnuje dva entropické kodéry, Huffmanův kodér a aritmetický kodér. Tato práce zpracovává další tři – ne zcela známou metodu 3R a dvě metody generující kódy s proměnlivou délkou – Golomb-Riceovo kódování a Fibonacciho kódování.

4.1 Recursive Range Reduction

Recursive Range Reduction (3R, také RRR) [4][12] je ve světě komprese poměrně neobvyklou metodou. Zatímco drtivá většina statistických kompresních algoritmů potřebuje značné množství vstupních dat k tomu, aby mohl kompresní model dosáhnout optimálního výkonu, kompresní poměr 3R je nezávislý na objemu vstupních dat. Základem je algoritmus RR (Range Reduction), který na vstupu přijímá seznam celých kladných čísel seřazený podle velikosti od největšího. Na takto neseřazená vstupní data je pak rekurzivně aplikován algoritmus 3R.

RR probíhá následovně: jako první se do výstupního proudu zakóduje hlavička, ve které je zapsána délka L bitové reprezentace L prvního (největšího) prvku. Zapsaná hodnota L bude o 1 nižší, protože libovolný počáteční prvek je reprezentovatelný vždy alespoň na jednom bitu. Hlavička má nastavenou pevnou velikost a dále se už její reprezentace nijak nekóduje. Za hlavičkou následující první hodnota seznamu se zakóduje na L-1 bitů bez předcházejících bitových nul a bez první bitové jedničky, která je implicitně předpokládána. Další číslo je zakódováno na L bitech, aby jej mohl dekodér jednoznačně rozpoznat. Pokud měla binární reprezentace čísla na nejvýznamnějších pozicích bitové nuly, bude o tento počet kratší následující zapsané číslo. Jelikož je seznam seřazený podle velikosti, je jisté, že jeho další prvek bude nižší, tzn. jeho binární reprezentace bude mít přinejmenším stejný počet předcházejících bitových nul.

Příklad: Pro zpracování RR máme dána čísla 179, 80, 62, 30, 2, 1. Algoritmus bude na výstup zapisovat následující data (uvažujme kódovanou velikost hlavičky 4 bity, aby pojala 16bitové hodnoty):

`0111` - (7) - první hodnota má 8 bitů (tedy L = 8)

`0110011` - (179) - zapíšeme jen 7 bitů, víme, že MSB je 1 a tak jej můžeme vynechat

0`1010000` - (80) - musíme zapsat L bitů, aby dekodér mohl rozpoznat číslo na vstupu

0`111110` - (62) - minulé číslo mělo 1 společnou předcházející nulu, můžeme vynechat

0`11100` - (30) - minulé číslo mělo 1 společnou předcházející nulu, můžeme vynechat

000`10` - (2) - minulé číslo mělo 1 společnou předcházející nulu, můžeme vynechat

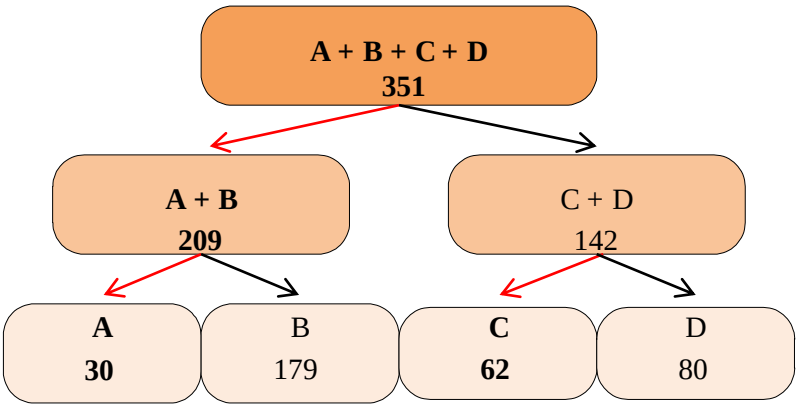
`01` - (1) - minulé číslo mělo 3 společné předcházející nuly, můžeme vynechat

Pokud budeme každý prvek výše uvedeného seznamu považovat za 8bitový, měla původní data délku 48 bitů. Po zpracování RR má výsledek 39 bitů. Jediná informace ukládaná mimo čísel je hlavička určující velikost prvního z nich, jelikož počet čísel, která budou zpracovávána, je předem znám.

V případě, že nemáme k dispozici předem seřazený seznam, použijeme rekurzivní variantu RR, ve které můžeme použít libovolnou posloupnost celých kladných čísel. Z čísel se vytvoří binární strom, ve kterém je kořenový uzel tvořen součtem všech těchto čísel, a listy jsou tvořeny jednotlivými čísly. Pomocí RR se zakódují průchody od kořene ke každému listu a kód každého mezikroku se

uloží do jeho uzlu. Potom stačí na výstup rekurzivně zakódovat jen polovinu celého stromu (kořen a buď jen pravé větve a listy nebo levé), zbytek hodnot se dá při dekódování dopočítat z těchto čísel.

Příklad: Pro zpracování pomocí 3R máme dána čísla 30, 179, 62, 80. Vytvoříme z nich perfektně vyvážený binární strom (viz obrázek 4.1) a pro každou větev vygenerujeme pomocí RR kódy. Poté stromem rekurzivně procházíme a na výstup posíláme kódy všech levých větví a listů.



Obrázek 4.1: Strom 3R koeficientů se zvýrazněnými zakódovanými hodnotami

4.2 Golomb-Riceovy kódy

Golomb-Riceovo kódování [10][12] je kombinace dvou různých kódování – Golombova a Riceova. Obě používají prefixový kód a jeden volitelný parametr, ve kterém se liší. Golombovo kódování používá parametr M = n (n ∈ N) a Riceovo kódování používá parametr M = n^2 (n ∈ N). Golomb-Riceovo kódování je nejvíce vhodné pro abecedy s geometrickým rozložením, tedy takovým, kde má několik nízkých čísel výrazně vyšší zastoupení než ostatní vyšší čísla, pro taková je jím tvořený prefixový kód optimální. Pro parametr `M` a kladné vstupní číslo `N` bude postup následující:

- číslo bude zapsáno v kódovém slově o dvou částech – kvocientu a zbytku
- kvocient `q` se vypočítá jako `N ÷ M`
- zbytek `r` se vypočítá jako `N mod M`
- první část kódového slova je tvořena `q` bitovými jedničkami a 1 bitovou nulou (unární kódování)
- u druhé části kódového slova záleží, jestli je parametr `M` mocnina dvou (`b = ⌈log2(M)⌉`)
    - pokud ano:
        - `r` se uloží beze změn na `b` bitech
    - pokud ne:
        - pokud `r < 2^b − M`, pak se `r` uloží na `b − 1` bitech
        - pokud `r ≥ 2^b − M`, pak se `r` uloží jako `r + 2^b − M` na `b` bitech

Příklad: Pro zakódování do Golomb-Riceova kódu o parametru M = 5 přichází číslo 13:

- `q = 13 ÷ 5 = 2`; do první části kódového slova zakódujeme 2 bitové jedničky a jednu nulu.
- `r = 13 mod 5 = 3`; parametr M = 5 není mocnina dvou.
- `b = ⌈log2(5)⌉ = 3`; `r ≥ 2^b − M`, do druhé části kódového slova zakódujeme 6 (`r + 2^b − M`) na 3 bitech. Výsledné kódové slovo pro vstupní hodnotu 13 je `110|110`.

4.3 Fibonacciho kódy

Fibonacciho kód [12] kladného celého čísla je jeho Fibonacciho reprezentace s jedničkou připojenou na konec. Například Fibonacciho kód pro 5 je `0001|1` a pro 33 je `1010101|1`. Použije se Fibonacciho posloupnost, jen redukovaná o první dva členy (0 a 1), tedy:
1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, 1597, 2584, 4181, 6765, 10946, 17711, …

Algoritmus kódování: V posloupnosti se najde nejvyšší možné číslo, po jehož odečtení od kódovaného čísla vyjde nezáporný výsledek. Výsledek se zakóduje na tolika bitech, na které pozici bylo toto číslo nalezeno. Poté se posloupností prochází zpět k začátku. Za každé číslo, které ještě lze odečíst, se předpojí bitová jednička, za každé číslo, které od mezivýsledku nelze odečíst, se předpojí bitová nula. Příklad pro číslo 7: od čísla 7 lze odečíst nejvýše číslo 5, které se v posloupnosti nachází na pozici 4, proto jeho kód bude uložen na 4 bitech. Odečte se číslo 5, mezivýsledek je 2, v bitovém řetězci je `1`. Další číslo v posloupnosti je 3, mezivýsledek je 2, nelze odečíst, v bitovém řetězci je `01`. Další číslo v posloupnosti je 2, mezivýsledek je 2, lze odečíst, v bitovém řetězci je `101`. Další číslo v posloupnosti je 1, mezivýsledek je 1, nelze odečíst, v bitovém řetězci je `0101`. Nakonec se k číslu připojí bitová jednička, aby jej mohl kodér jednoznačně rozeznat. Výsledný kód pro 7 je tedy `01011`.

Algoritmus dekódování: Bitový řetězec se načítá tak dlouho, dokud nebudou načteny dvě bitové jedničky po sobě. Poté se poslední bitová jednička odstraní. Pak se postupně prochází řetězcem a za každou jedničku se k výsledku přičte číslo, které je ve Fibonacciho posloupnosti na stejné pozici. Příklad pro vstupní řetězec `010011`: Odtrhne se jednička, řetězec je `01001`. Jedničky jsou na druhé a páté pozici. Přičtou se tedy hodnoty z druhé a páté pozice Fibonacciho posloupnosti – 2 a 8. Zakódované číslo na vstupu je 10.

Ukázka prvních několika čísel a jejich Fibonacciho kódů je v tabulce 4.1:

1 11	2 011	3 0011	4 1011	5 00011	6 10011
7 01011	8 000011	9 100011	10 010011	11 001011	12 101011

Tabulka 4.1: Ukázka Fibonacciho kódů prvních 12 čísel

5 Implementace a srovnání

V jazyce C++ byl vytvořen program IBP implementující navrhovaná vylepšení standardu. Program je kompilovatelný pomocí frameworku GCC na platformách Windows a *nix. Zdrojové kódy jsou přiloženy na CD.

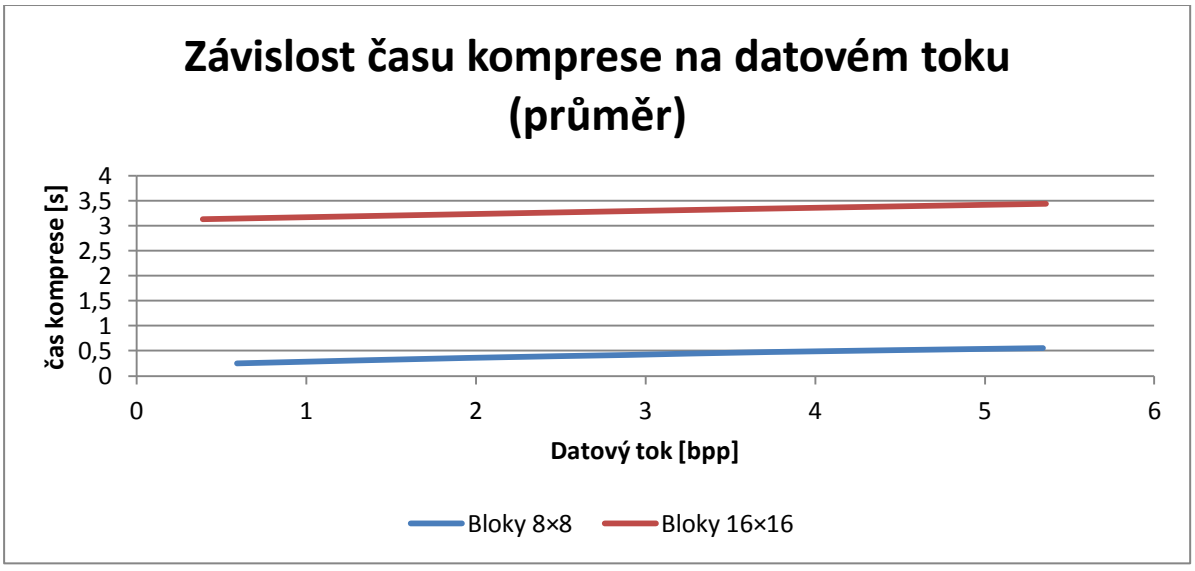
Pro provedení testů byly použity standardní testovací obrázky Lenna, Baboon a obrázek papoušků ze sady bezztrátových obrazů Kodak [9]. Dále byl použit obrázek generovaný počítačem pomocí metody ray tracing [14] a snímek obrazovky operačního systému. Zpravidla je použita pouze průměrná hodnota výsledků všech pěti obrázků. Testovací obrázky jsou 24bitové.

Na srovnání kvality komprimovaných obrazů byly použity metriky PSNR (Peak Signal-to-Noise Ratio – špičkový odstup signálu od šumu) a SSIM (Structural Similarity – strukturální podobnost).

Pokud není uvedeno jinak, v následujících srovnáních je použito nastavení shodné se standardem, tj. Huffmanův kodér a linearizační průchod cik-cak. V případě zobrazení dvou grafů závislosti kvality na datovém toku sleduje levý graf vývoj hodnot PSNR, zatímco v pravém jsou vyneseny hodnoty SSIM.

5.1 Srovnání času komprese obou velikostí bloků

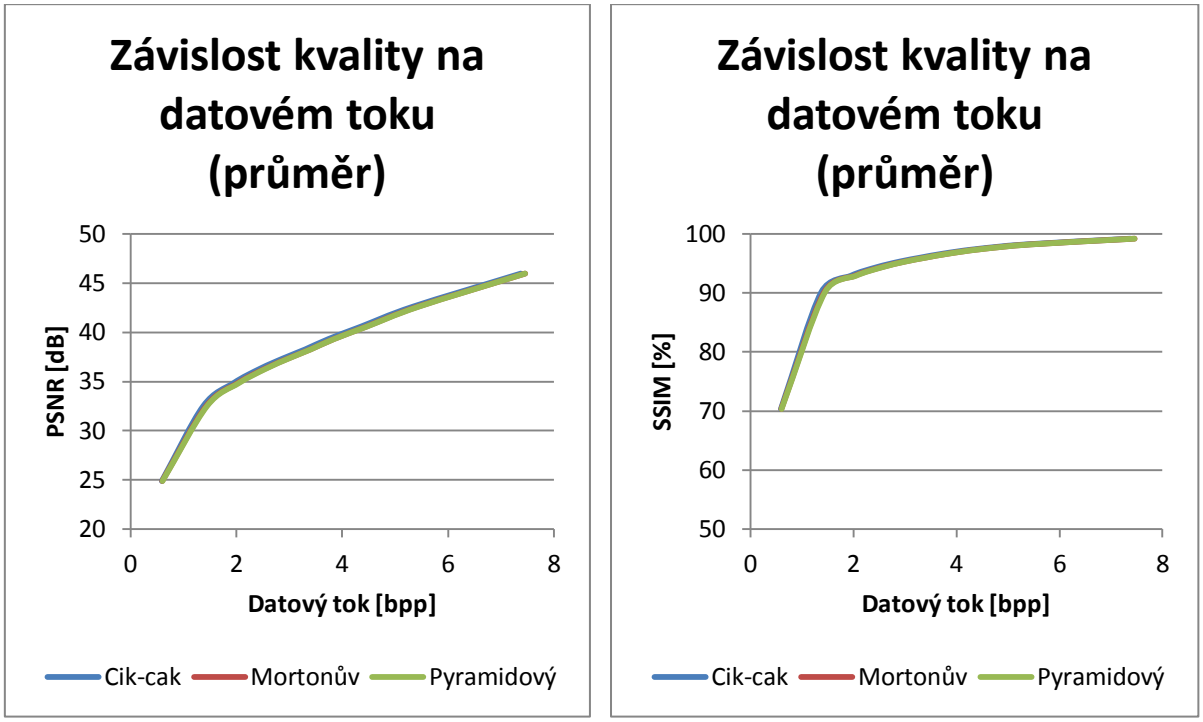
Srovnání bylo provedeno s kodérem 3R a linearizačním průchodem cik-cak. Pro srovnání byly použity standardní tabulky (u bloků velikosti 16×16 lineárně roztažené).



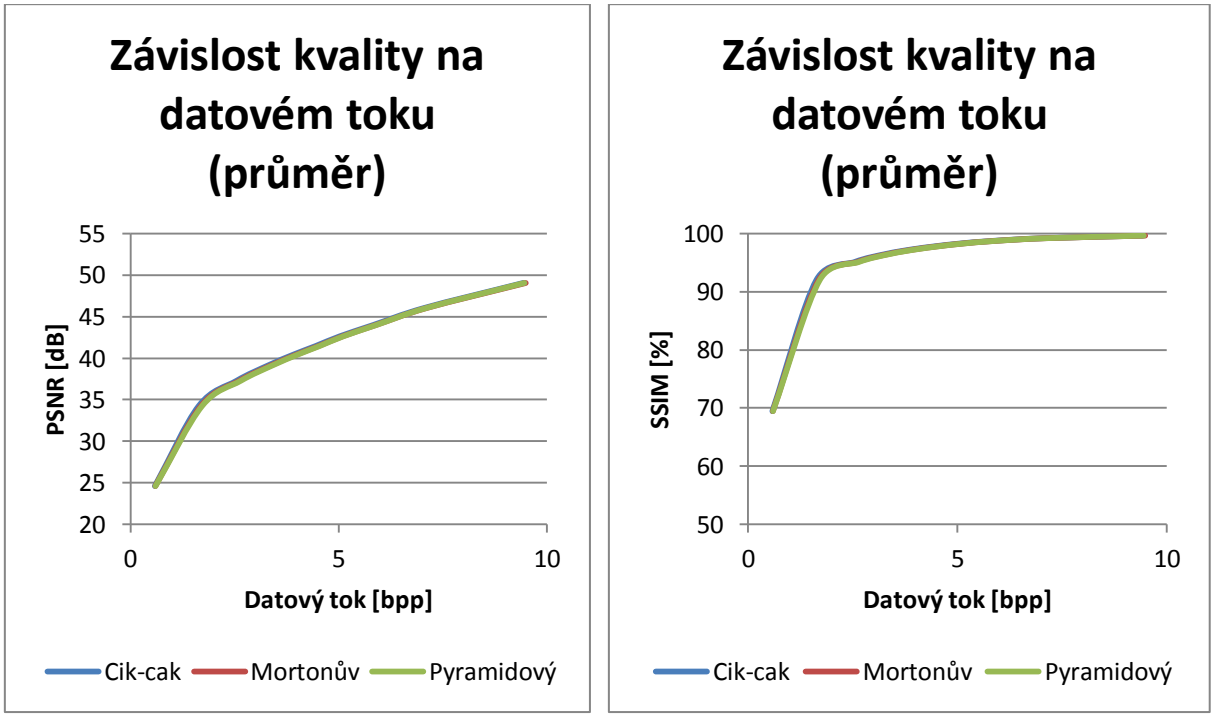
Graf 5.1: Závislost času komprese na datovém toku

Jak ilustruje graf 5.1, po navýšení velikosti bloku došlo k mnohonásobnému zpomalení komprese i přesto, že se s nárůstem doby zpracování jednoho bloku zároveň snížil počet zpracovávaných bloků.

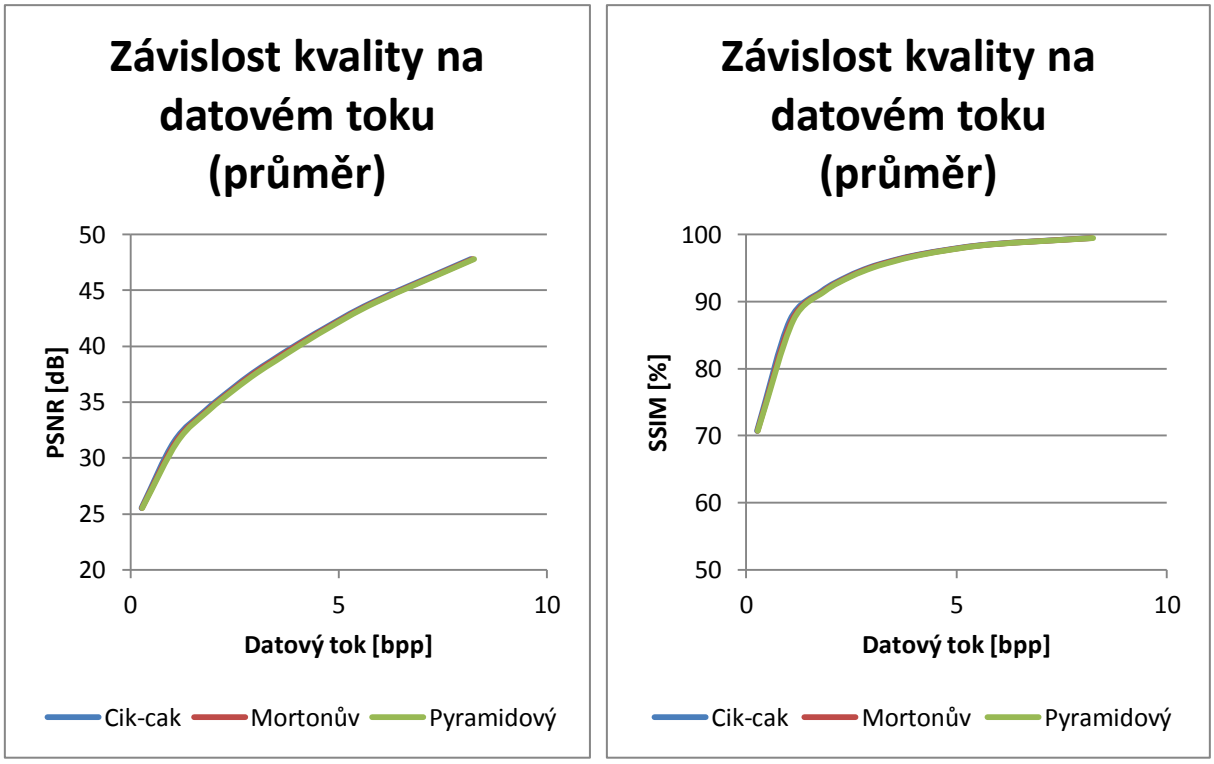
5.2 Srovnání linearizačních průchodů



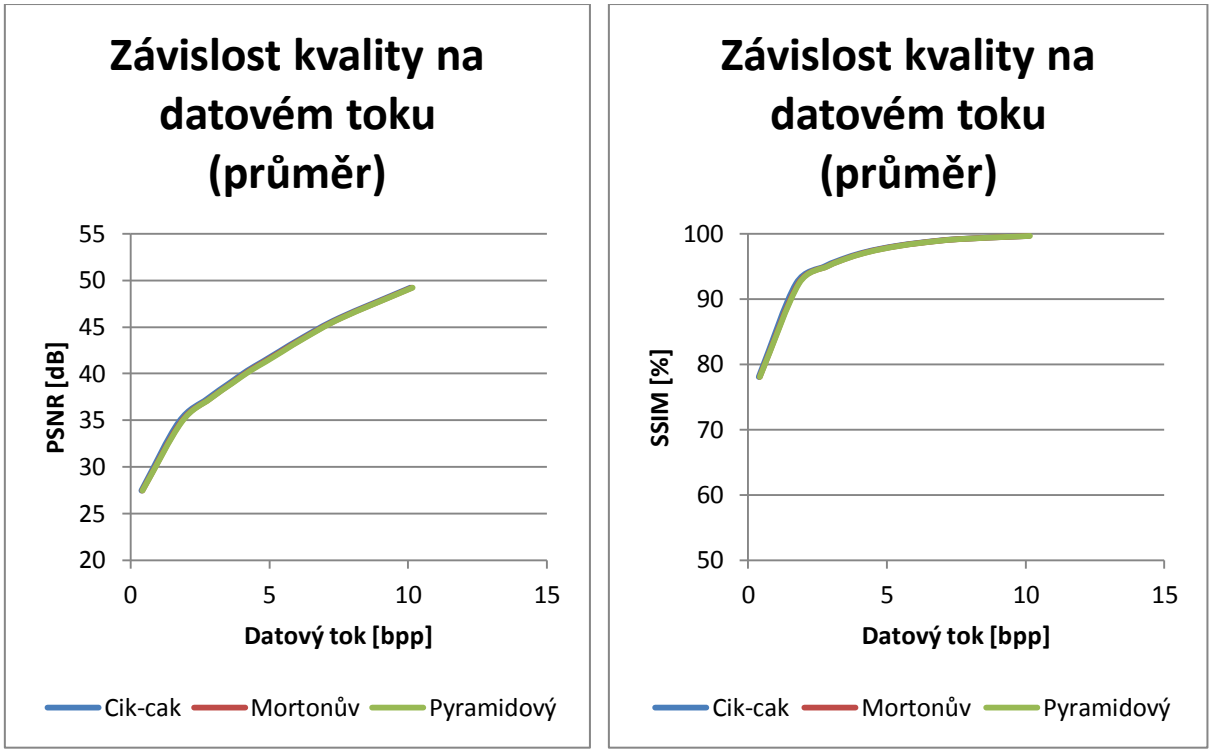
Graf 5.2: Srovnání linearizačních průchodů pro bloky 8×8 a standardní kvantizační tabulky



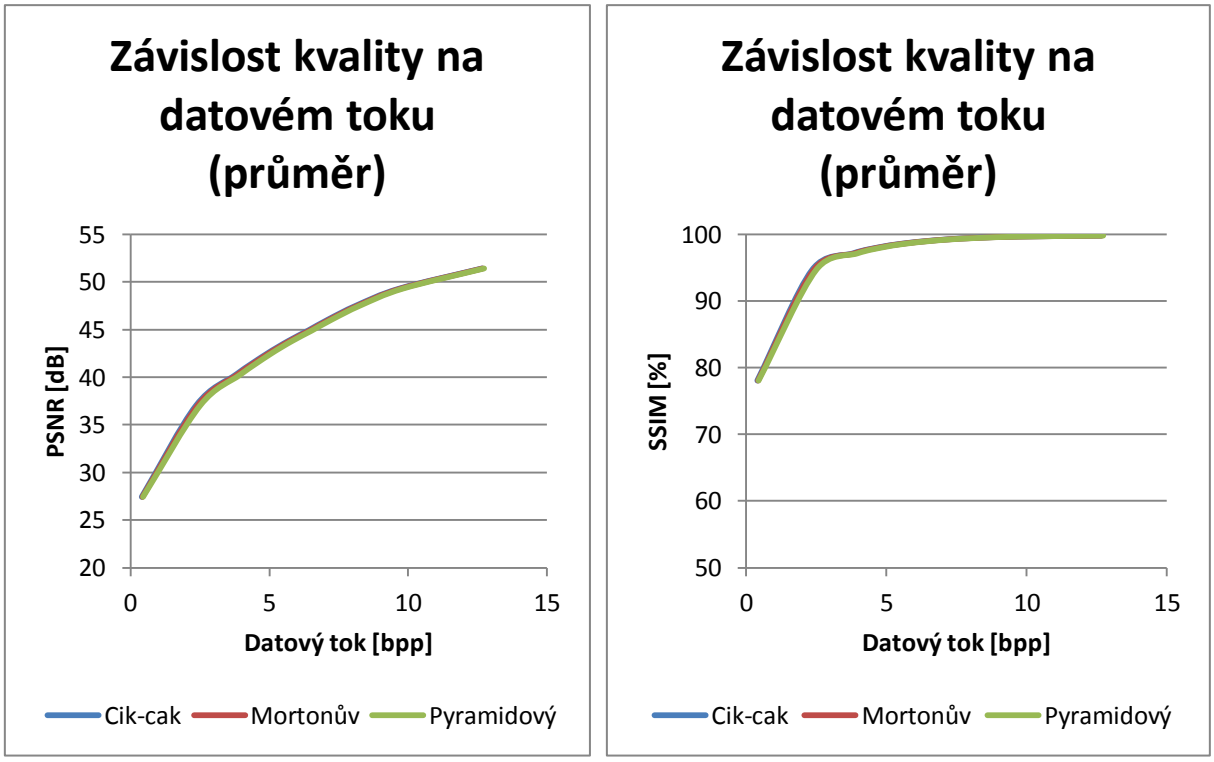
Graf 5.3: Srovnání linearizačních průchodů pro bloky 8×8 a HVS kvantizační tabulky



Graf 5.4: Srovnání linearizačních průchodů pro bloky 16×16 a standardní kvantizační tabulky doplněné maximy



Graf 5.5: Srovnání linearizačních průchodů pro bloky 16×16 a lineárně roztažené standardní kvantizační tabulky



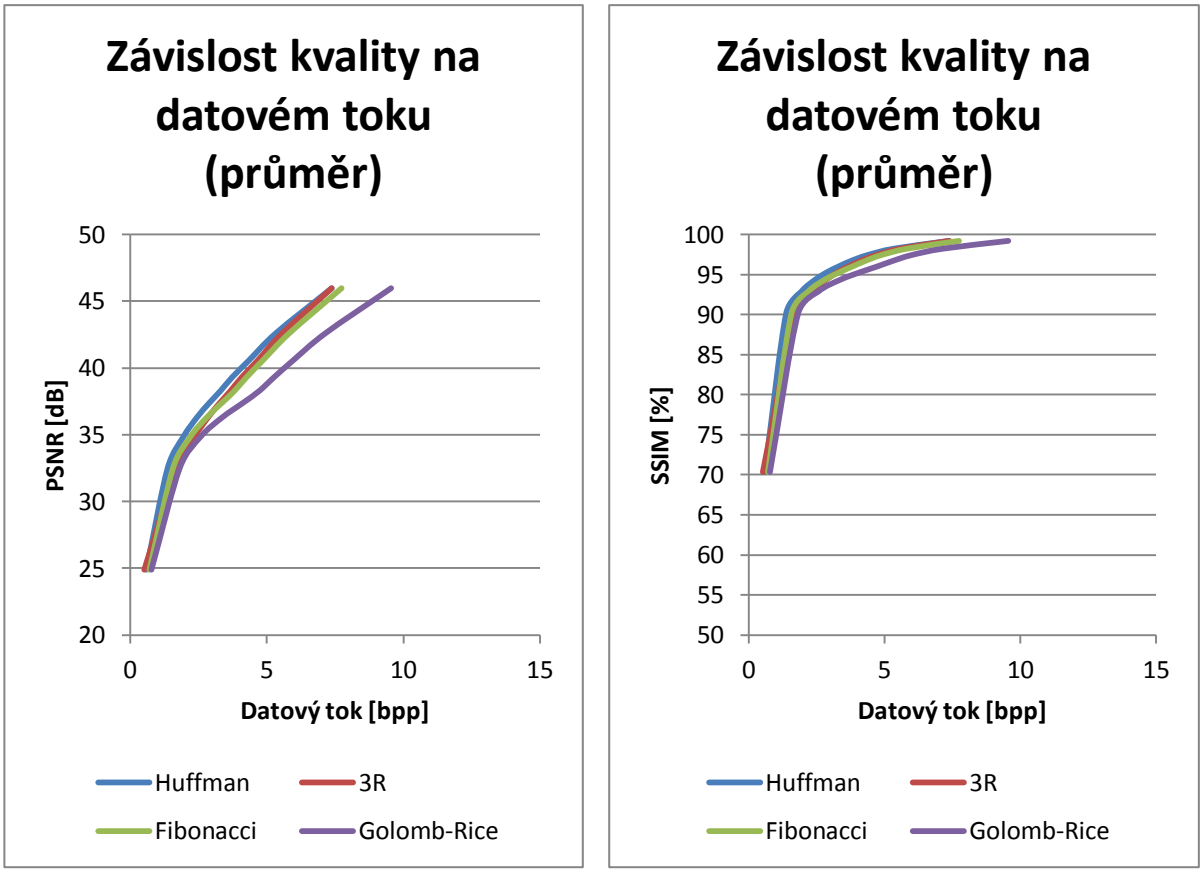
Graf 5.6: Srovnání linearizačních průchodů pro bloky 16×16 a lineárně roztažené HVS kvantizační tabulky

Jak je možné vidět na grafech 5.2 až 5.6, libovolná volba velikosti bloků a kvantizačních tabulek se na výkonu Huffmanova kodéru projevila zanedbatelně. Cik-cak průchod je nejúčinnější, po něm následuje Mortonův průchod a pyramidový průchod je v porovnání nejslabší. Mezi jednotlivými průchody jsou však v tomto srovnání naprosto minimální rozdíly.

5.3 Srovnání entropických kodérů

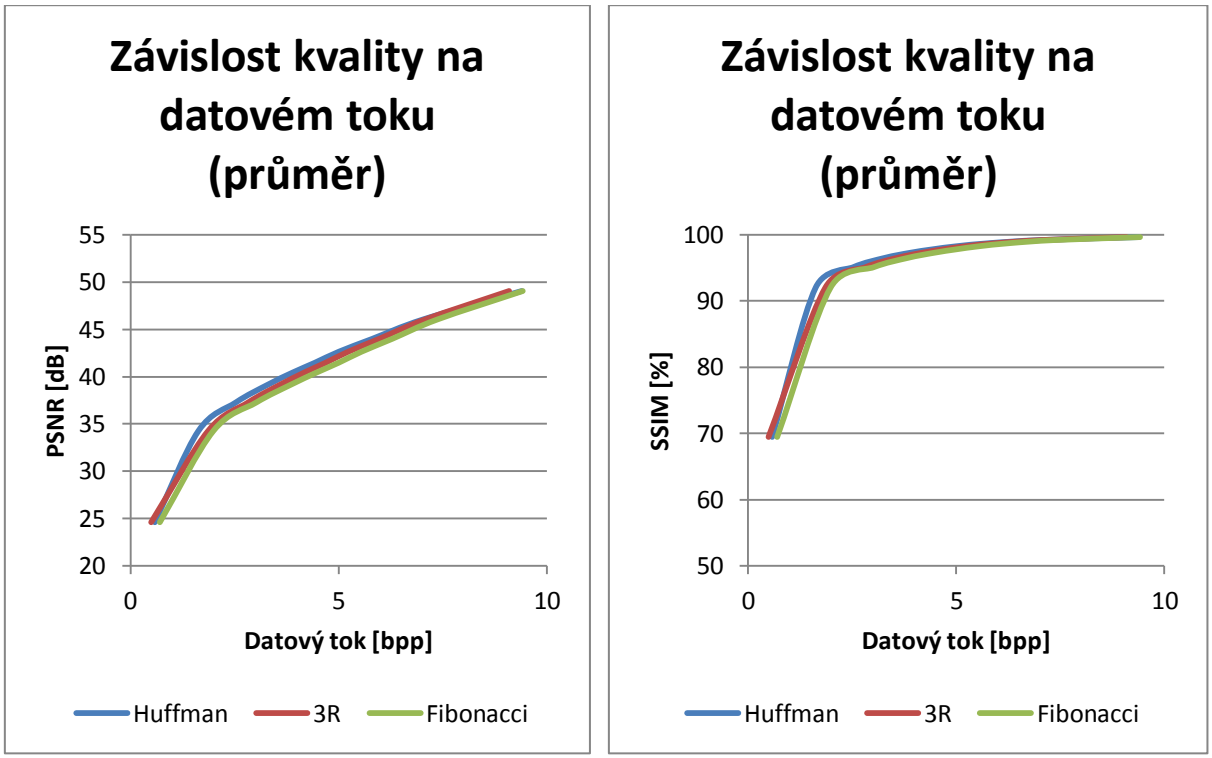
Jelikož nestandardní kodéry přirozeně nezahrnují specifické postupy pro kódování DC a AC koeficientů, použil jsem u Fibonacciho a Golomb-Riceova kodéru následující optimalizace: na DC koeficient je aplikována DPCM transformace a je zakódován Huffmanovým kodérem. Poté je uložen index posledního nenulového AC koeficientu a konkrétním kodérem zakódované všechny AC koeficienty až po tento index. Pro metodu 3R byla aplikována pouze DPCM transformace DC koeficientů. U které metody to bylo třeba, byla záporná čísla převedena na kladná (kladná čísla vratně namapována na sudá čísla, záporná na lichá).

Huffmanův kodér je naimplementován dle JPEG standardu, Golomb-Riceův kodér, Recursive Range Reduction a Fibonacciho kodér jsou navrhovaná vylepšení. Každé srovnání obsahuje dvojici grafů sledujících kvalitativní metriky. Levý graf sleduje PSNR a pravý graf SSIM. Následující srovnání jsou provedena s použitím standardního cik-cak průchodu.

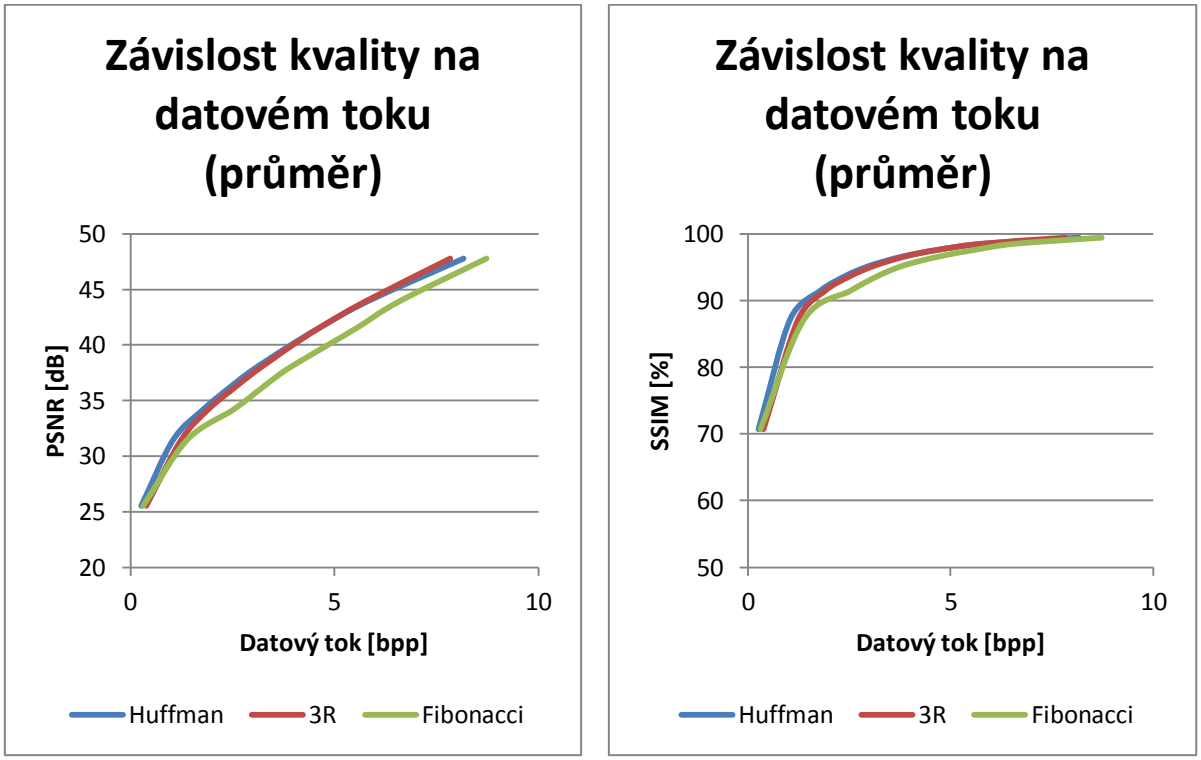


Graf 5.7: Srovnání entropických kodérů pro bloky 8×8 a standardní kvantizační tabulky

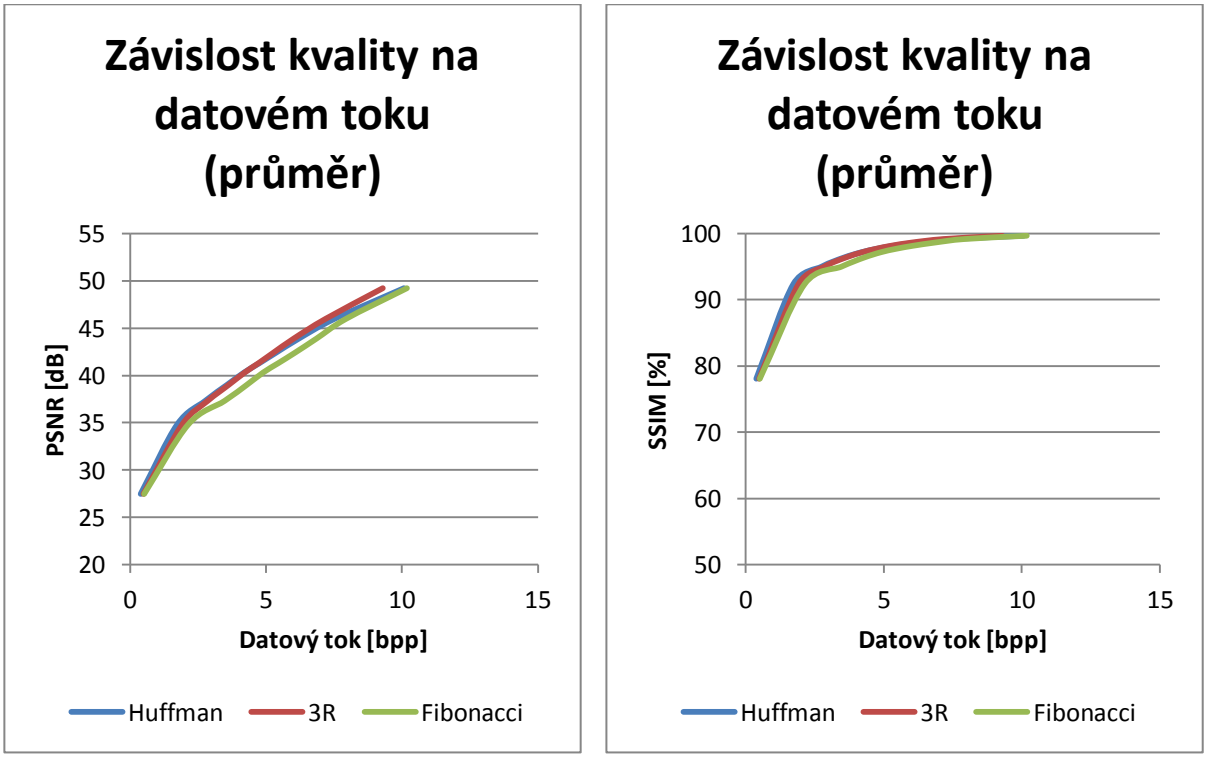
Vzhledem ke zřetelně nízké účinnosti Golomb-Riceova kodéru v tomto i ostatních testovaných případech byl tento z dalšího srovnávání entropických kodérů vynechán.



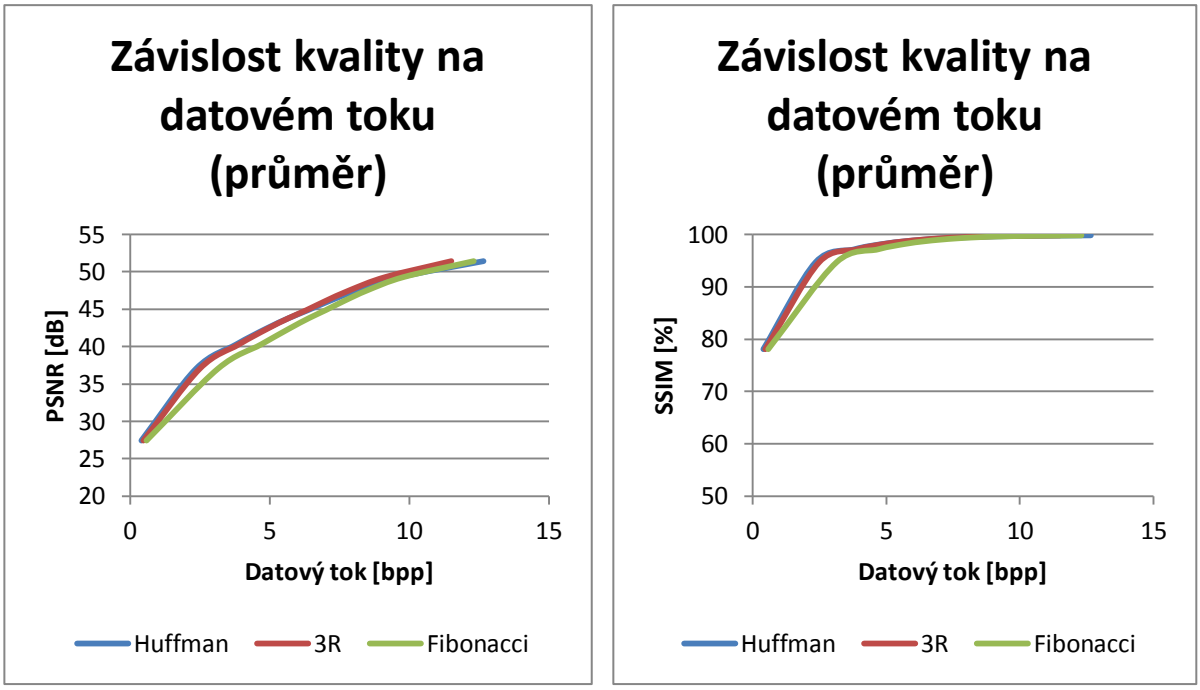
Graf 5.8: Srovnání entropických kodérů pro bloky 8×8 a HVS kvantizační tabulky



Graf 5.9: Srovnání entropických kodérů pro bloky 16×16 a standardní kvantizační tabulky doplněné maximy



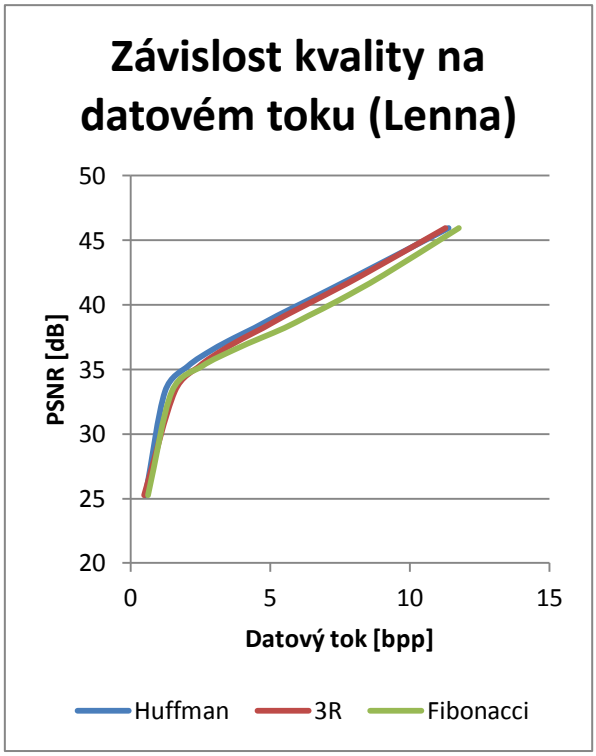
Graf 5.10: Srovnání entropických kodérů pro bloky 16×16 a lineárně roztažené standardní kvantizační tabulky

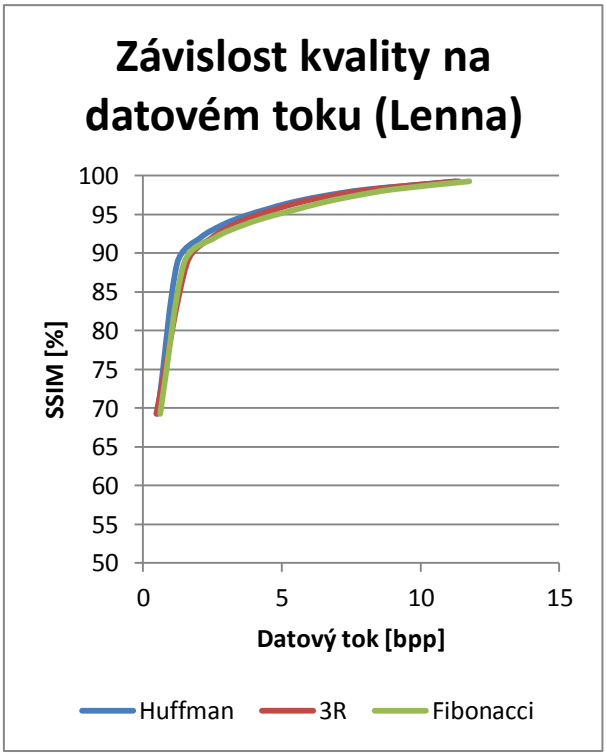


Graf 5.11: Srovnání entropických kodérů pro bloky 16×16 a lineárně roztažené HVS kvantizační tabulky

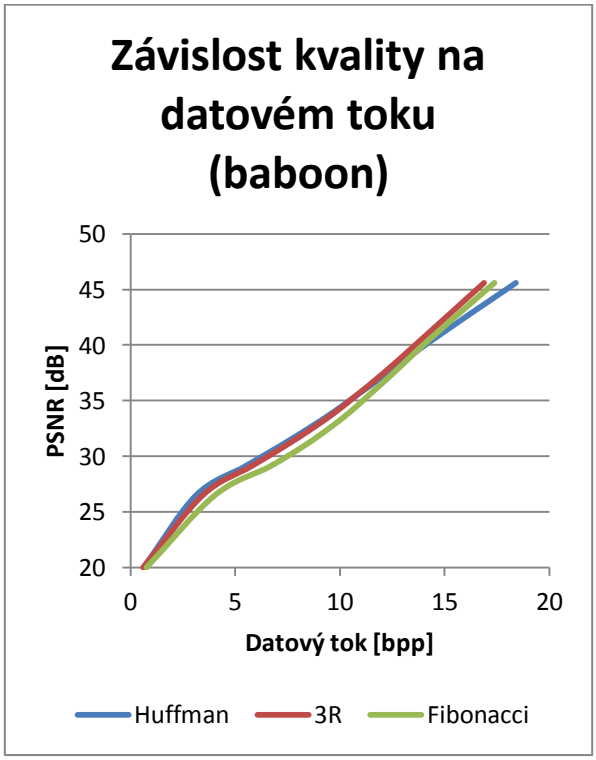
5.3.1 Vliv jednotlivých obrázků na souhrnných hodnotách

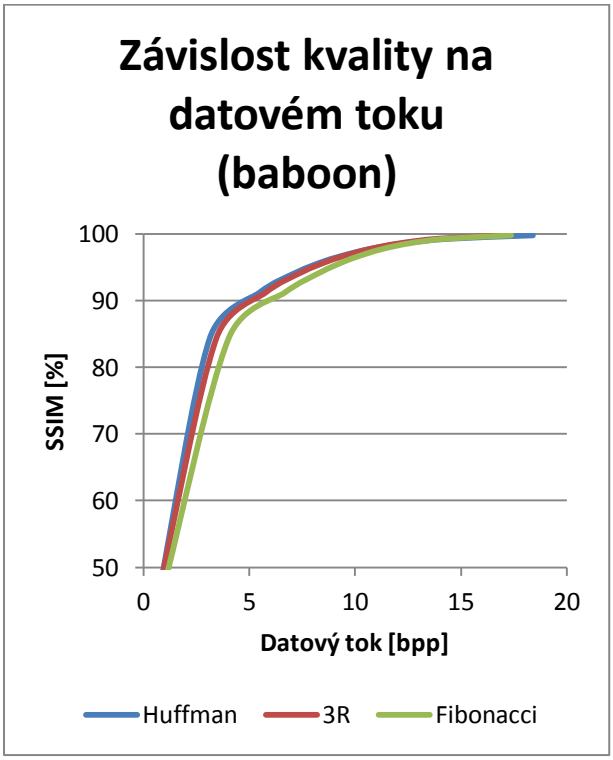
V tomto srovnání jsou uvedeny výsledky entropických kodérů pro bloky 8×8 a HVS kvantizační tabulky, tentokrát pro každý obrázek samostatně. Ač je u každého z měření vidět odlišný trend, průměr měření pěti obrázků přináší poměrně vyrovnané výsledky. Můžeme pozorovat, že v obrázcích *baboon* a *ReactOS* metoda 3R překonává Huffmanovo kódování na vysokých datových tocích, ač se to nedá tak jednoznačně říct o průměrných hodnotách z celé testovací sady. Metoda 3R také překonává Huffmanovo kódování na nejnižších datových tocích.



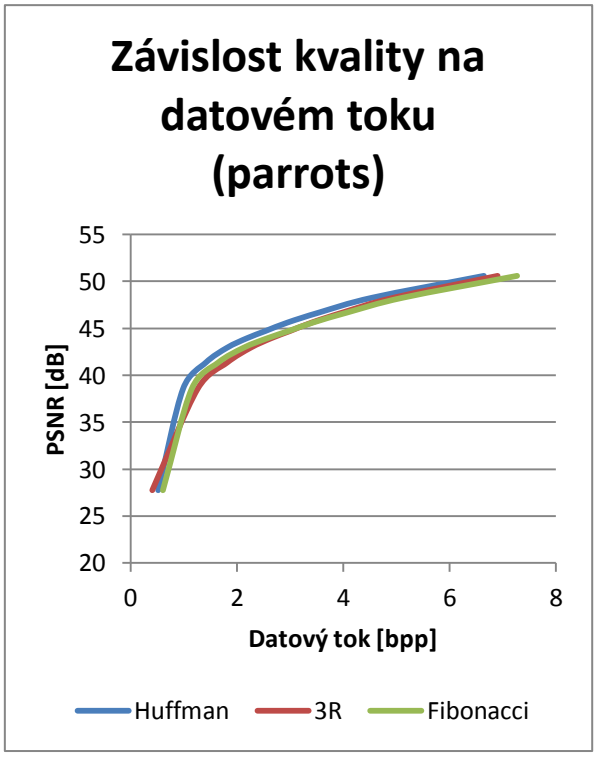


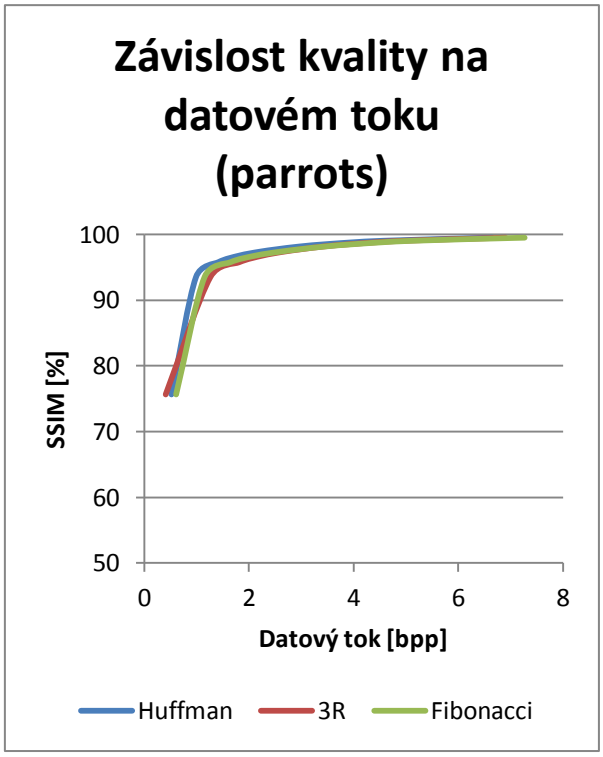
Graf 5.12: Srovnání entropických kodérů na obrázku Lenna pro bloky 8×8 a HVS kvantizační tabulky



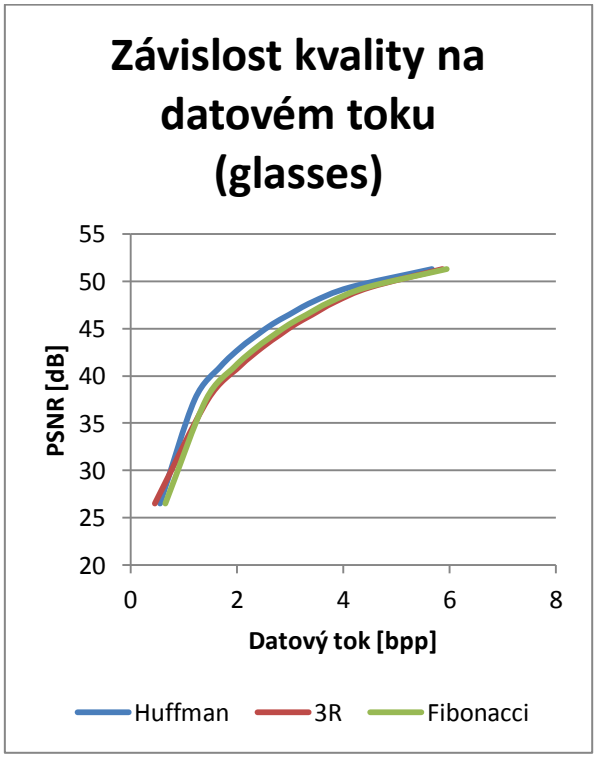


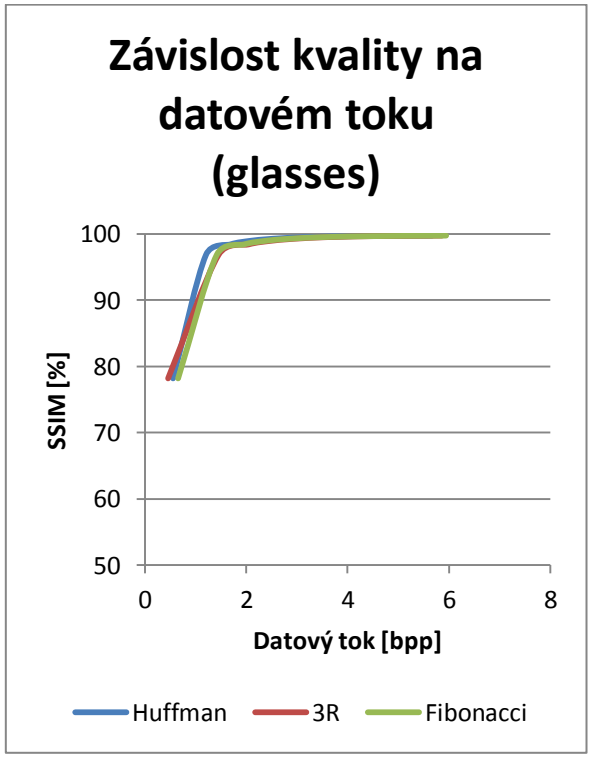
Graf 5.13: Srovnání entropických kodérů na obrázku baboon pro bloky 8×8 a HVS kvantizační tabulky



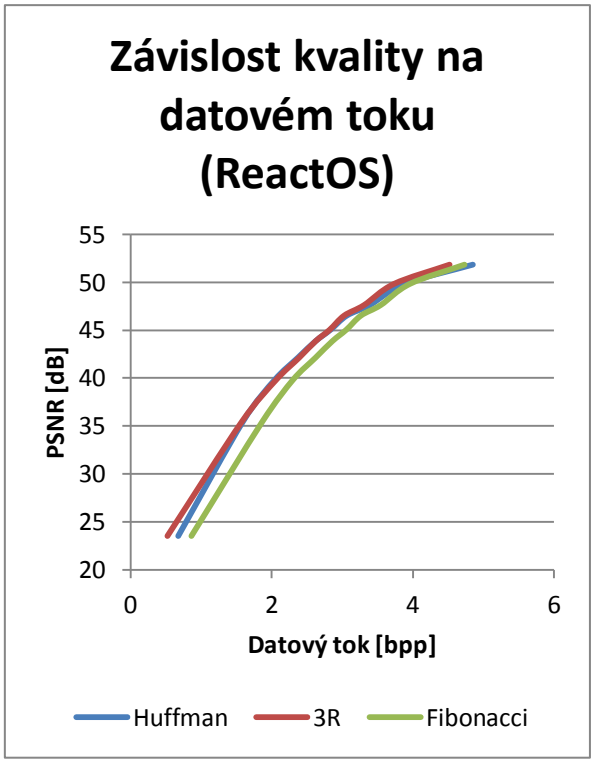


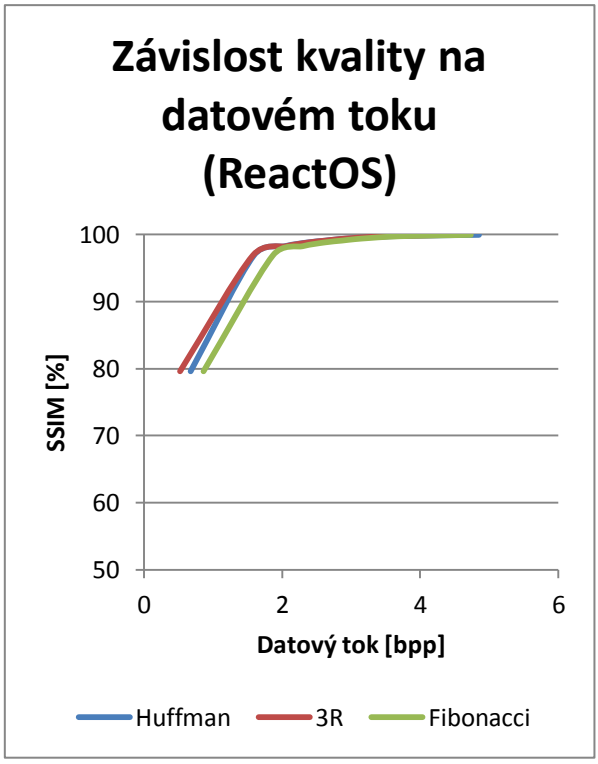
Graf 5.14: Srovnání entropických kodérů na obrázku „parrots“ pro bloky 8×8 a HVS kvantizační tabulky



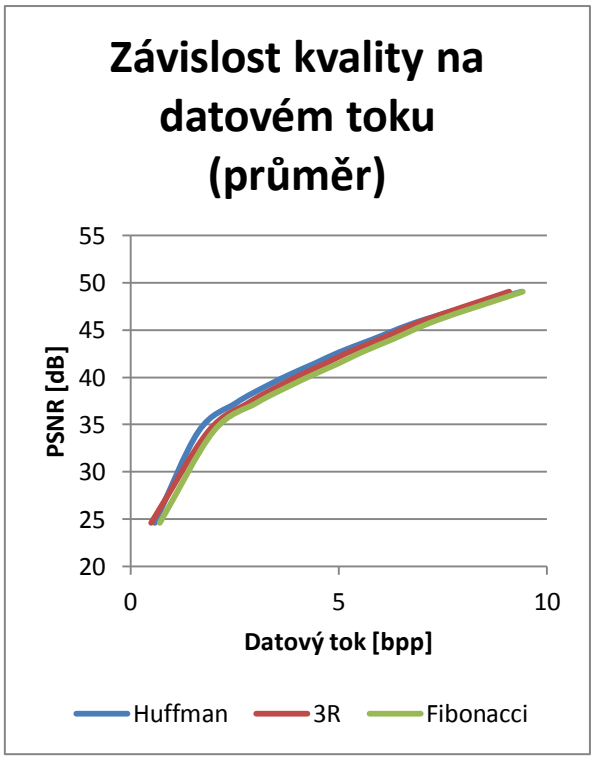


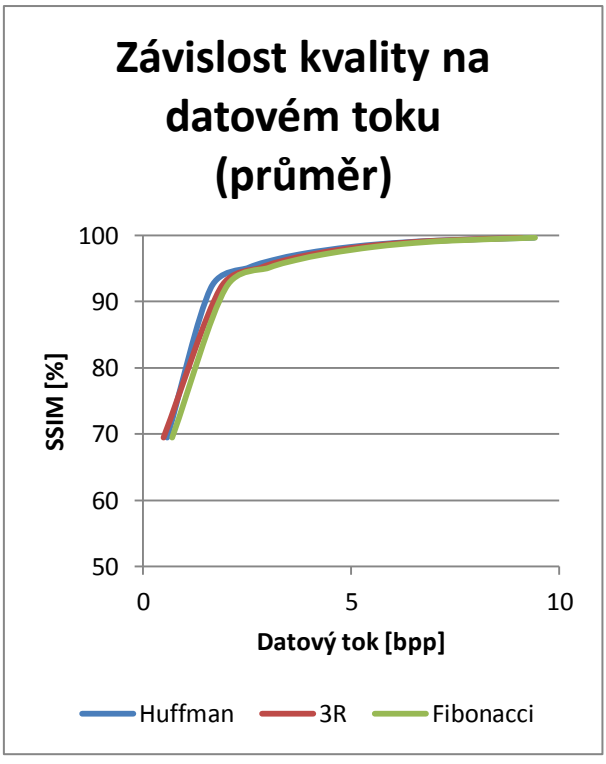
Graf 5.15: Srovnání entropických kodérů na obrázku glasses pro bloky 8×8 a HVS kvantizační tabulky





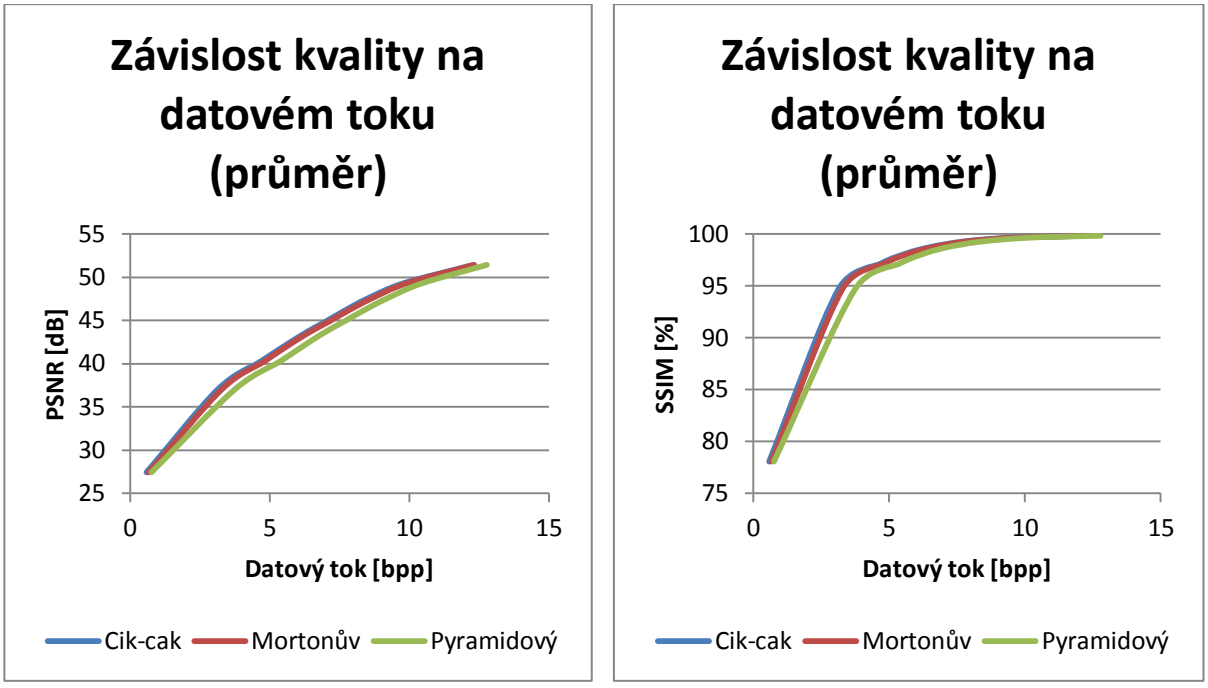
Graf 5.16: Srovnání entropických kodérů na obrázku „ReactOS“ pro bloky 8×8 a HVS kvantizační tabulky



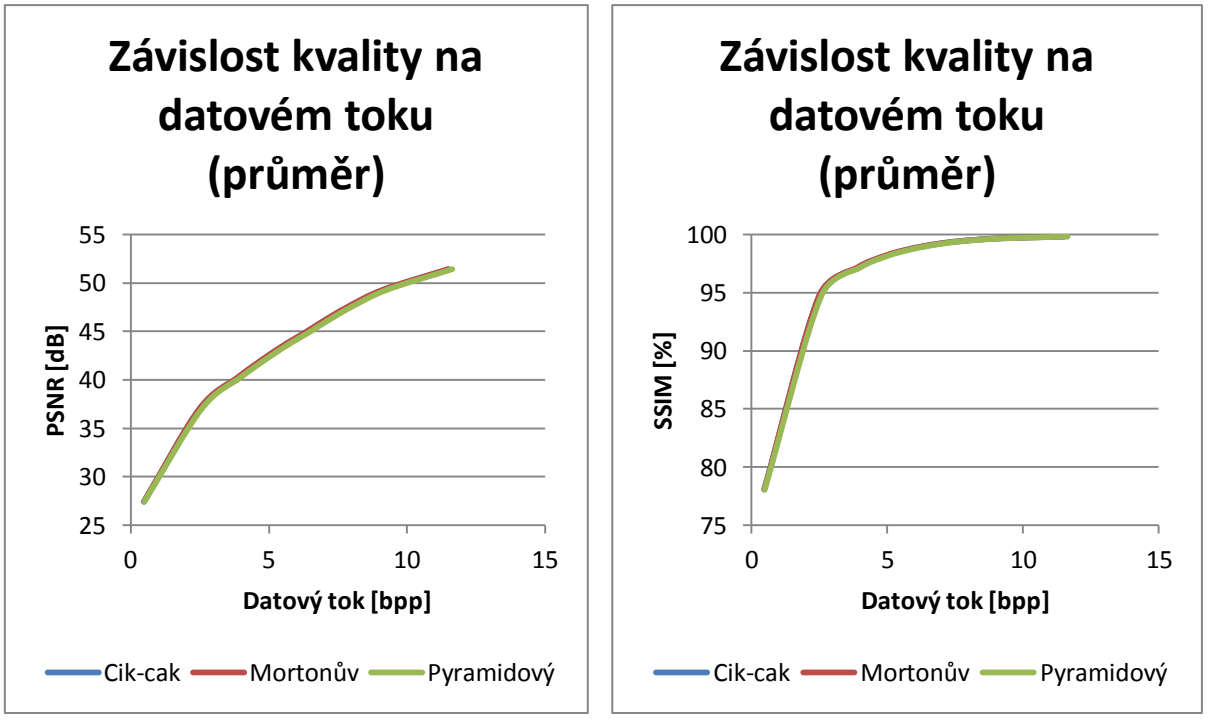


Graf 5.17: Srovnání entropických kodérů pro bloky 8×8 a HVS kvantizační tabulky

5.4 Srovnání kombinací vylepšení

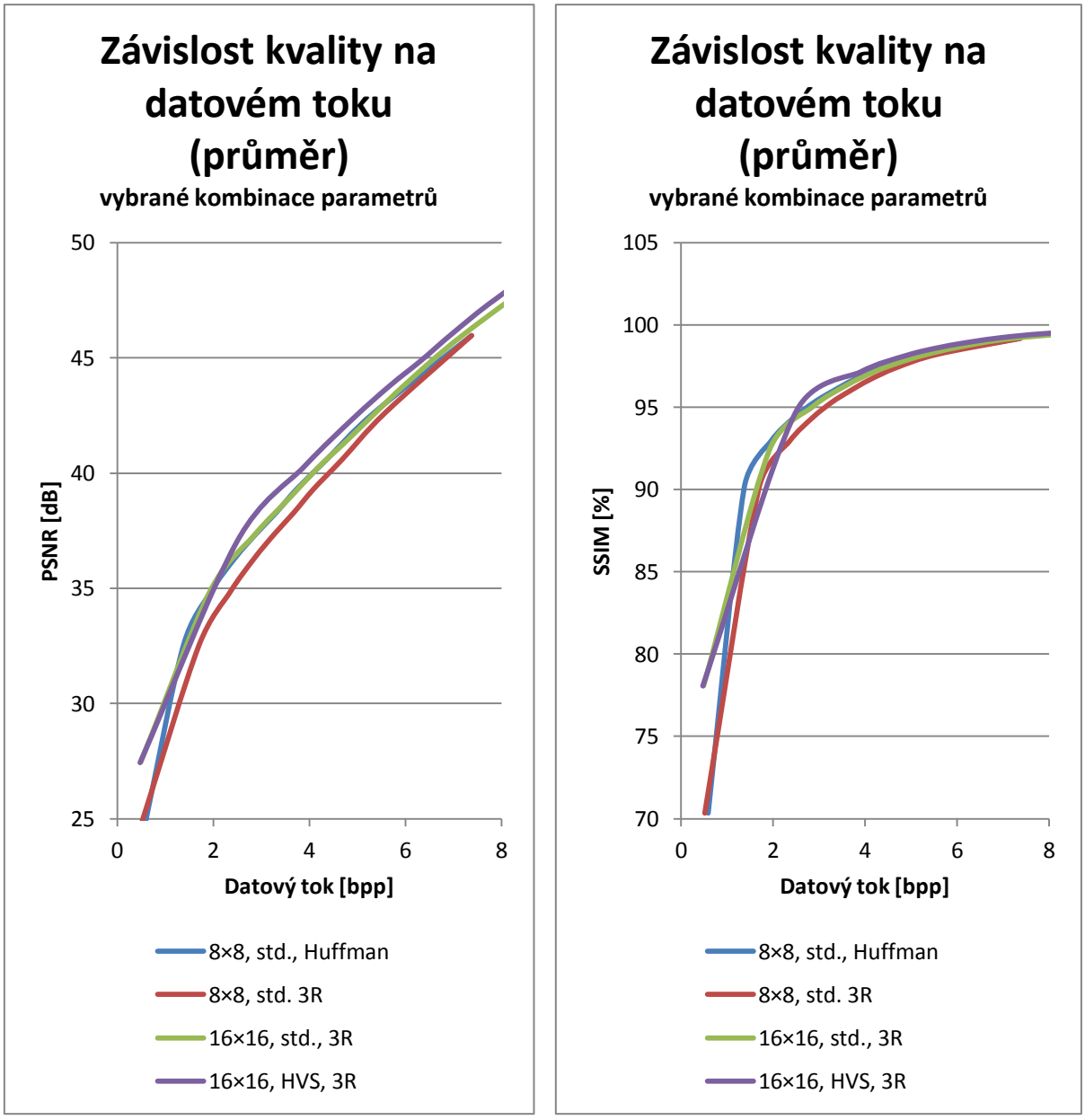


Graf 5.18: Srovnání vlivu linearizačních průchodů na Fibonacciho kodér pro bloky 16×16 a lineárně roztažené HVS kvantizační tabulky



Graf 5.19: Srovnání vlivu linearizačních průchodů na kodér 3R pro bloky 16×16 a lineárně roztažené HVS kvantizační tabulky

Grafy 5.18 a 5.19 dokazují, že volba pyramidového průchodu má negativní na všechny kodéry. Výkon 3R (viz grafu 5.19) zůstává téměř nezměněný stejně jako výkon Huffmanova kodéru.



Graf 5.20: Srovnání vybraných nastavení velikostí bloku, kodéru a kvantizačních tabulek

Na grafu 5.20 lze vidět, že použitím vylepšených kvantizačních tabulek, vyšší velikosti bloku a nestandardního algoritmu lze dosáhnout lepšího kompresního poměru než Huffmanův kodér.

6 Závěr

V práci byly prozkoumány existující techniky a jejich kombinace ve ztrátové kompresi obrazu běžně neužívané, a byl zjištěn jejich vliv na účinnost komprese. Srovnání navrhovaných zlepšení ukazují, že cik-cak průchod je ze srovnávaných linearizačních průchodů nejefektivnější v přesouvání nenulových koeficientů na začátek datového proudu tak, aby se linearizovaný blok dal ukončit symbolem EOB co nejdříve a nekódovalo se více dat, než je nezbytně nutné. Navrhovaný Mortonův a pyramidový průchod jej nepřekonávají. Srovnání také ukazuje, že 3R je jediný z testovaných kodérů, který dokáže konkurovat Huffmanovu kodéru a dokáže jej ve většině datových toků při vhodném nastavení překonat.

6.1 Přínos

Během tvorby této práce byl vytvořen příspěvek do soutěže STUDENT EEICT 2011, byl obhájen před komisí a přijat do sborníku. Sborník STUDENT EEICT 2011 je dostupný na WWW: <http://www.feec.vutbr.cz/EEICT/2011/sbornik/>, příspěvek lze najít v části Bakalářské projekty, sekce Grafika a multimédia.

Zdrojové kódy programu mohou být díky objektovému návrhu rozšířeny o další linearizační průchody a entropické kodéry, jsou šířeny pod svobodnou licencí GPL. Zdrojové kódy jsou dostupné z WWW: <http://ibp.black-fox.cz/>

Byly provedeny další experimenty s transformacemi dat mezi kvantizačním a linearizačním krokem. První testovaná transformace bylo delta kódování nejen DC koeficientů, ale odděleně i na AC koeficientech. Další pokus byl proveden s Move-To-Front transformací [3] AC koeficientů. Move-To-Front je reverzibilní transformace, která vytváří pole o délce rovné nejvyššímu z posloupnosti nezáporných prvků jí zpracovávaných. Každý prvek pole má hodnotu rovnou jeho indexu. Poté se prochází posloupností a hodnota daného prvku je vyhledána v poli a přesunuta na jeho začátek. Všechny hodnoty, které se nacházely před ní, jsou v poli odsunuty o vzdálenost tohoto jednoho prvku. Touto transformací lze u několika často se opakujících libovolných čísel dosáhnout posloupnosti nízkých čísel, což může vylepšit efektivitu komprese. Následně bylo experimentováno s dělením různě velkých podmnožin AC koeficientů různými konstantami. Vliv všech těchto experimentů na kompresní poměr při zachování kvalitativních metrik byl negativní a nebyl do práce zakomponován.

6.2 Další vývoj

Jako další rozšíření této práce by bylo možné implementovat lokální kosinovou transformaci [1]. Tento koncept umožňuje do určité míry předejít artefaktům, které při JPEG kompresi s vysokým kompresním poměrem vznikají na hranici bloků a v některých případech působí téměř až dojmem mřížky položené přes obraz. Lokální kosinová transformace se při kompresi provádí před DCT a při dekompresi po inverzní DCT. Tvoří doplňkovou transformaci a nijak nezasahuje do jádra JPEG kodéru, aplikuje se mimo něj. Při stejné úrovni komprese její autoři dosáhli vyšších hodnot PSNR a obrázky zpracované touto transformací byly v subjektivních testech hodnoceny jako kvalitnější než obrázky, na které transformace nebyla aplikována.

Dále plánuji dosáhnout kompatibility s podmnožinou JPEG standardu – baseline sequential – základní sekvenční postup. V současné podobě práce a za předpokladu, že bude použit Huffmanův kodér a cik-cak linearizační průchod, je k vytvoření kompatibility s baseline sequential nutné použití JFIF[1] segmentů, alespoň SOI, APP0/APP1, DQT, SOF0, DHT, SOS a EOI. Je na zvážení, jakým způsobem by se mohl obrázek vytvořený mým kodérem pomocí nestandardních technik upravit, aby s ním dokázal standardní dekodér alespoň částečně pracovat. Lze například uložit miniaturu obrázku ve standardní podobě, aby bylo bez mého nestandardního dekodéru možné zobrazit alespoň jeho náhled, a komprimovaná data uložit ve standardem nepodporovaném segmentu.

V programu jsou implementovány a srovnávány bloky o velikosti 8×8 a 16×16, kód je v současné podobě nutné upravit, aby bylo možné testovat míru zlepšení i pro velikosti bloku 32×32, případně 64×64. Vyšší velikost bloků už by pravděpodobně nebyla využitelná kvůli extrémnímu nárůstu kompresního i dekompresního času.

[1] JFIF = JPEG File Interchange Format

Seznam zkratek

JPEG	skupina, která tento kodek vytvořila (*Joint Photographic Experts Group*)
DCT	diskrétní kosinová transformace (*Discrete Cosine Transform*)
DC	stejnosměrná složka (*Direct Current*)
AC	střídavá složka (*Alternating Current*)
DPCM	rozdílové kódování (*Differential Pulse-Code Modulation*)
RLE	*Run-Length Encoding*
EOB	konec bloku (*End Of Block*)
ZRL	běh nul (*Zero Run-Length*)
MPS	pravděpodobnější symbol (*More Probable Symbol*)
LPS	méně pravděpodobný symbol (*Less Probable Symbol*)
3R	*Recursive Range Reduction*
RR	*Range Reduction*
PSNR	špičkový poměr signálu k šumu (*Peak Signal to Noise Ratio*)
SSIM	strukturální podobnost (*Structural Similarity*)
VLC	kód s proměnnou délkou (*Variable Length Code*)
JFIF	Formát kontejneru pro uložení JPEG dat (*JPEG File Interchange Format*)

Literatura

[1] AHARONI, G; AVERBUCH, A.; COIFMAN, R.; ISRAELI, M. Local Cosine Transform – A method for the reduction of the blocking effect in JPEG. In *Journal of mathematical Imaging and Vision, Special Issue on Wavelets, Vol. 3*. Houten: Springer Netherlands, 1993. s. 7-38.

[2] AHMED, N.; NATARAJAN, T.; RAO, K. R. Discrete Cosine Transform. In *IEEE Transactions on Computers*. New York: Institute of Electrical and Electronics Engineers, 1974. s. 90-93.

[3] BENTLEY, J. L.; SLEATOR, D. D.; TARJAN, R. E.; WEI, V. K. A Locally Adaptive Data Compression Scheme. In *Communications of the ACM – Vol. 29, No. 4*. New York: Association for Computing Machinery, 1986. s. 320-330.

[4] GUIDON, Y. *Data compression: the "3R" algorithm* [online]. 2007 [cit. 2011-04-25]. Dostupné z WWW: <http://ygdes.com/ddj-3r/ddj-3r_compact.html>.

[5] HUFFMAN, D. A. A Method for the Construction of Minimum-Redundancy Codes. In *Proceedings of the I.R.E*. New York: The Institute of Radio Engineers, Inc., 1952. s. 1098–1102.

[6] CHANG, L. W; WANG, C. Y.; LEE, S. M. Designing JPEG quantization tables based on human visual systém. In *International Conference on Image Processing ICIP '99*. New York: Institute of Electrical and Electronics Engineers, 1999. s. 376-380.

[7] *Impulse Adventure: JPEG Chroma Subsampling* [online]. c2011 [cit. 2011-05-10]. Dostupné z WWW: <http://www.impulseadventure.com/photo/chroma-subsampling.html>.

[8] ISO/IEC 10918-1:1994. *Information technology – Digital compression and coding of continuous-tone still images: Requirements and guidelines*. Geneva: International Organization for Standardization, 1994. 182 s.

[9] *Kodak Lossless True Color Image Suite* [online]. c2011 [cit. 2011-05-10]. Dostupné z WWW: <http://r0k.us/graphics/kodak/>

[10] MAHONEY, M. *Data Compression Explained* [online]. [s.l.] Dell, Inc., 2011 [cit. 2011-04-29]. Dostupné z WWW: <http://mattmahoney.net/dc/dce.html>.

[11] MORTON, G. M. *A computer Oriented Geodetic Data Base; and a New Technique in File Sequencing*, 1966. Ottawa: IBM Canada Ltd.

[12] SALOMON, D.; MOTTA, G. *Handbook of Data Compression*. 5th ed. London: Springer, 2009. 1383 s. ISBN 978-1848829022.

[13] VOLLBEDING, G. *ITU-T JPEG-Plus Proposal for Extending ITU-T T.81 for Advanced Image Coding*. Geneva: Independent JPEG Group, 2006. Dostupné z WWW: <http://jpegclub.org/temp/ITU-T-JPEG-Plus-Proposal_R3.doc>.

[14] *Wikimedia Commons* [online]. c2011 [cit. 2011-04-03]. Dostupné z WWW: <http://commons.wikimedia.org>.

[15] *Wikipedie: Otevřená encyklopedie: Mortonův rozklad* [online]. c2011 [cit. 2011-04-30]. Dostupné z WWW: <http://cs.wikipedia.org/w/index.php?title=Morton%C5%AFv_rozklad&oldid=6444830>.

Seznam příloh

Příloha 1. Manuál programu

Na přiloženém CD jsou zdrojové kódy testovacího programu ve složce `src` a zkompilované binární soubory pro Microsoft Windows ve složce `bin`. Ke kompilaci programu je třeba použít kompilátor g++, v systému musí být nainstalována knihovna libGD.

Pro nastavení použití bloků o velikosti 16×16 a pro změnu výběru kvantizační tabulky je třeba odkomentovat či zakomentovat vybrané nastavení v řádcích 30-40 souboru `common.h` a program překompilovat.

Použití programu:

```
ibp <operace> vstup vystup [-q kvalita] [-c kodér] [-l linearizátor]
```

Parametry:

	<*operace*>: `c` (komprese), `d` (dekomprese), `e` (porovnání dvou obrázků)
(volitelný)	<*kvalita*>: od `1` do `100`
(volitelný)	<*kodér*>: `1`=Huffmanův, `3`=Golomb-Riceovy kódy, `4`=3R, `5`=Fibonacciho kódy
(volitelný)	<*linearizační průchod*>: `1`=cik-cak, `2`=Mortonův, `3`=pyramidový

Výchozí parametry:

```
-q 50 -c 4 -l 1
```

Příklad komprese a dekomprese:

```
ibp c image.png test.out -q20 -c3
ibp d test.out image_2.png
```

Příklad porovnání:

```
ibp e image.png image_2.png
```

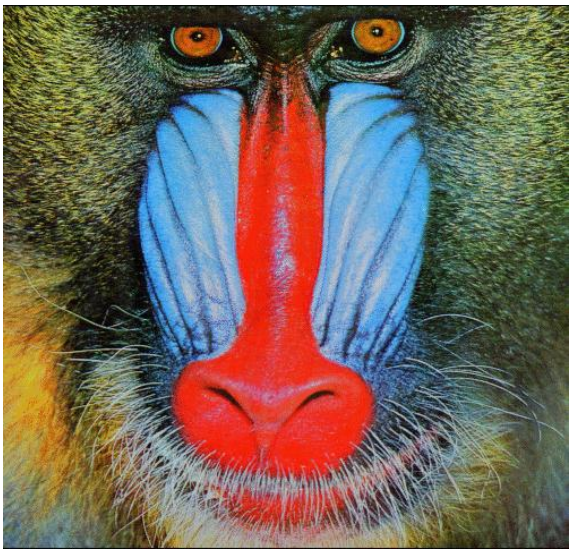
Varianty programu:

`ibp_8_std` – bloky 8×8, standardní kvantizační tabulky
`ibp_8_hvs` – bloky 8×8, HVS kvantizační tabulky
`ibp_b16_std-padded` – bloky 16×16, doplněné standardní kvantizační tabulky
`ibp_b16_std-interpolated` – bloky 16×16, roztažené standardní kvant. tabulky
`ibp_b16_hvs` – bloky 16×16, HVS kvantizační tabulky

Příloha 2. Testovací obrázky

K testování byly použity tyto obrázky, jejich bezztrátové verze jsou přiloženy na CD ve složce *images*.

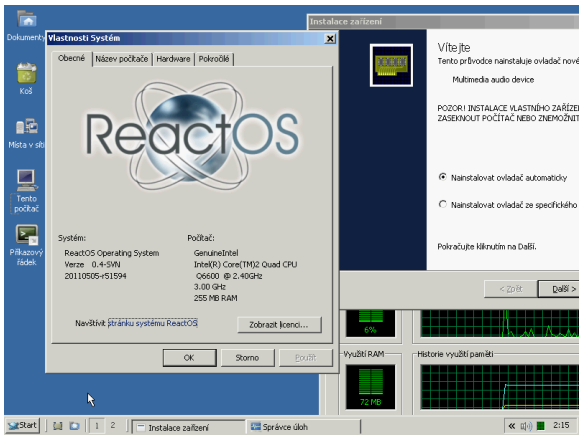
Obrázek 6.1.a: Lenna



Obrázek 6.1.b: Baboon

Obrázek 6.1.c: Parrots

Obrázek 6.1.d: Glasses



Obrázek 6.1.e: ReactOS

Příloha 3. CD